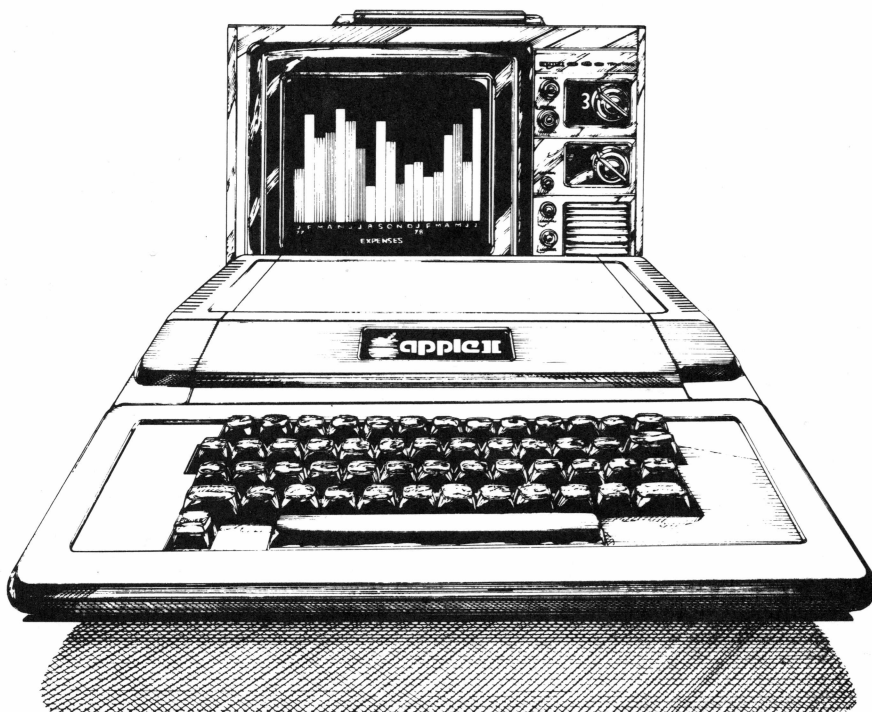
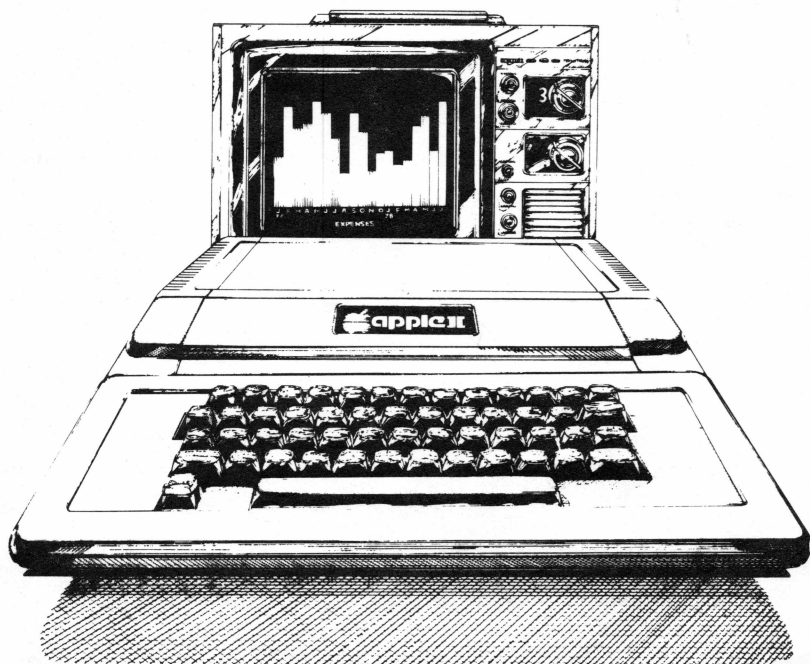


Computer Station'sTM Programmer's Handbook to the Apple II[®]



Apple, Apple II and Applesoft are the registered trademarks of
Apple Computer, Inc.

to the
Apple II



Produced by
Computer Station
#12 Crossroads Plaza
Granite City, Illinois 62040

Edited & Compiled by
Conway B. Christensen

©Copyright 1981 by
Computer Station's, Inc.

Product no.
7702

The Apple® Pascal system incorporates UCSD Pascal™ and Apple extensions for graphics and other functions. UCSD Pascal was developed largely by the Institute for Information Science at the University of California at San Diego, under the direction of Kenneth L. Bowles.

"UCSD PASCAL" is a trademark of The Regents of The University of California. Use thereof in conjunction with any goods or services is authorized by specific license only and is an indication that the associated product or service has met quality assurance standards prescribed by the University. Any unauthorized use thereof is contrary to the laws of the State of California.

The APPLE CP/M Digital Research section has been reproduced with the permission of copyright owner,
Digital Research, Inc., Pacific Grove, CA,
modified by author.

VisiCalc computer program and reference card
copyright 1979, 1980 Software Arts, Inc. Reference
card reprinted by permission of Software Arts, Inc.

PREFACE

This book is intended as a quick reference manual for programmers. It does not replace the all important manuals but, it does reduce the need of carrying all those manuals with you for reference once the language or program is known.

This booklet is not meant as a replacement for these excellent publications of APPLE COMPUTER INC., rather, it is designed as a companion. The information contained in the APPLE publications could not be condensed into one booklet this size. Instead, the command syntax, operands, and special locations have been gleaned from the reference manuals and collected to yield a Programmer's reference. We assume the programmer has already read and absorbed APPLE's publication. This programmer's guide is designed as a helpful tool for speedy information retrieval without referring to the entire text found in the APPLE publications.

APPLE REFERENCE MANUALS USED IN PREPARING THIS BOOK

1. APPLE II — The DOS Manual
Apple Product #A2L0036
(030-0115-00)
2. APPLESOFT II
#A2L006
(030-0013-03)
3. APPLE 6502 — Assembler/Editor
#A2L00
(030-0112-00)
4. APPLE II — APPLE PLOT
#A2L0037
(030-0116-00)
5. APPLE II — APPLESOFT TOOL KIT
#KA2L0038
(030-0113-00)
6. APPLE WRITER
7. APPLE II — REFERENCE MANUAL
#A2L0001A
(030-0004-01)
8. DOS 3.2 Version
#A2L0012
(030-0011-01)
9. PASCAL
#A2L0027
(030-0101-00) AND #A2L0028
(030-0100-00)

NON-APPLE REFERENCE MANUALS USED IN PREPARING THIS BOOK

1. VISICALC

Personal Software Inc.
1330 Bordeaux Drive
Sunnyvale, CA 94086

MICROSOFT Z-80 SOFTCARD Vol. 1&2

Microsoft Consumer Products
400 108th Ave. N.E. Suite 200
Belleville, Washington 98004

THE CP/M HANDBOOK WITH MP/M

by Rodney Zaks
SYBEX INC. PUBLICATION

MISC. REFERENCE MATERIAL

SUPER DISK COPY III

Sensible Software
6619 Perham Drive
West Bloomfield, Michigan 48033

PROGRAMMER'S HANDBOOK UTILITY DISK

(Available as Optional Item)

CODE — POKES WRITER

A section of memory in program lines inclusion in a program

HELLO 2

Menu driven xx diskette

We wish to express special thanks to:

PHYLLIS COLE of Apple Computer, Inc.
JEFF WALDEN of Personal Software, Inc.
DOTTIE HALL of Microsoft

DAN BRICKLIN of Software Arts
DOROTHY McEWEN and
PHILLIP NELSON of Digital
Research, Inc.

for their cooperation in the preparation of the Computer Station Programmers Handbook to the Apple II.

Respectfully,
LYNN K. BUSBY
Pres., Computer Station's, Inc.

INDEX TO COMPUTER STATION'S PROGRAMMER'S HANDBOOK

SECTION

PAGE

MONITOR

System Monitor Commands	1
Examine Memory	1
Change Memory	1
Move Memory	1
Verify Memory	2
Cassette I/O	2
Display	2
Dis-Assembler	2
Mini-Assembler	2
Monitor Program Execution and Debugging	3
Hexidecimal Arithmetic	3
Set Input / Output Ports	4
Multiple Commands	4
Special Control and Editing Characters	4

DOS

Disk Operating System	1
Bootting DOS	1
Initializing Slave Diskettes	1
Changing Slave Diskettes to Master Diskettes	1
Listing Files on Diskette	1
Disk Copy Programs	2
Apple Copy Programs	2
Copy From DOS 3.2 to DOS 3.3	2
FID-Copy Single Files DOS 3.3	2
Other Copy Programs	3
Disk Copy / Disk Space in ROM	3
Running 13 Sector Disks in 16 Sector System	3
Disk Commands	4
General Program Control	4
Machine Code	4
Files	5
Control	5
ROM / RAM Applesoft	6
Memory Maps	7
Memory Maps Over-Written When Booting DOS	7
Memory Areas Used by DOS and Either Basic	8
Memory Areas Over-Written When Booting DOS	9
Memory Areas Used by DOS and Either BASIC	10
Himem Value Set by Booting DOS	11
DOS Entry Points	11

ASSEMBLER EDITOR

Apple 6502 Assembler Editor	1
Command Syntax	1
Syntax of Parameter Lists	2
Editing Commands	3
Disk and Tape Commands	4
Operating Commands	5
Editor	6

BASICS

Applesoft Extended, Floating-Point Basics	1
Simple Variables	1
Array Variables	1
Algebraic Operators	1
Relational and Logical Operators	1
System and Utility Commands	2
Editing and Format-Related Commands	2
Arrays and Strings	3
Commands Relating to Flow & Control	4
Graphics and Game Controls	5
Some Math Functions	6
Applesoft Basic Command Index	7
High-Resolution Operating Subroutines	8
Decimal Tokens For Keywords	9
ONERR GOTO Codes	10
Using Trace From Applesoft & Its Affects With DOS	11
Reserved Words in Applesoft & Integer Basic	11
Applesoft ROM Card	13
Algebraic Operators	13
Relational and Logical Operators	13
Applesoft Hires Routines	14
Zero Page Locations	14
The Graphics Routines	14
Applesoft Error Messages	14
Error Codes	15
Commands Related to Errors	15
Disk Error Messages	16
Applesoft Floating Point Format	16
Somek Applesoft Internal Routines	17
RE: False I/O Read Strokes	19
Applesoft Commands and Functions	19
Command Subroutines in the Cassette Version	22

PASCAL

System Information	1
Volume Names	1
File Names	1
Recommended Suffix Conventions For Filenames	1
File Specification	1
The Workfile	2
Commands Available At All Levels	2
Command Level	3
Filer	4
Filer Commands	5
Editor	5
Editor Commands	6
Compiler	8
System Functions and Procedures	8
Syntax Errors in Apple UCSD Pascal	11
Assembler	14
Assembler Directive Summary	14
Assembler Syntax Errors	16
Linker	17
Utilities	18
Utilities Summary	18
Execution Error Messages	20
I/O Error Messages	22
Device Volume Numbers and Names	23
Apple I/O Device Slots	23
Reserved Words in Pascal	24
Default Parameters for System Miscinfo	25
Format and Disk Copy Commands	26
P-Machine Op-Codes	27
Pascal Fast Floating-Point Processor Intrinsic Routines	28

MACRO-SCED

Macro-Screen Editor	1
Macro-Sced Editor Table of Contents	2
Summary of Features	3
Activating the Screen Editor	3
Summary of Commands	5
Output Controls	6
The Keyboard Macro Function	8
Standard Macro Table	11

UTILITIES

Applesoft Programmer's Assistant	1
High Resolution Graphics	2
Summary of Animatrix Commands	3

WORD PROCESSORS

Apple Writer	1
Modes	1

Extraordinary Characters	2
Printer Commands	3
Appendix D: ASCII Characters	3
VISICALC	
Visicalc Reference Cards	1
Moving the Cursor	1
The ESC Key	1
Setting A Label Entry	2
Setting A Value Entry	2
Value References	2
Functions	2
Examples of Functions	3
Entry Ranges	4
Lists	4
Commands	4
Printing	5
Replicate	6
Storage Commands	6
CP/M	
Apple CP/M Digital Research	1
Booting Apple CP/M	1
To Format a Disk	2
To Copy System Tracks 0-2 To New Disks	1
To Copy Full Disk	1
General Operational Notes For CP/M Operating System	2
New Keyboard Characters	2
Keyboard Editing	2
Output Control	2
Changing Disks & CP/M Warm Boot	2
File Name Specifiers	2
Built in CP/M Commands	2
Program Descriptions	3
Command Editing Controls	4
Extension Types	4
Transient Commands	5
ED CP/M Editor	7
Control Characters	7
Editor Commands	7
Editor Conditions	10
ASM.COM CP/M Assembler	10
Program Format	10
Assembler Directives	10
Error Messages	11
DDT.COM CP/M Dynamic Debugging Tool	11
Editing	11
Command	11
PIP.COM CP/M Peripheral Interchange Program	12

Response to '*'	12
Keywords	13
PIP Parameters	14
Device Names	16
DUMP.COM	16
LOAD.COM	16
SUBMIT.COM	16
CP/M Error Messages	17
BDOS ERR on d: BAD SECTOR	17
BDOS ERR on d: SELECT	17
BDOS ERR on d: READ ONLY	17
BDOS ERR on d: FILE R/O	17
Function Summary	18
BASIC 80	
Microsoft Basic-80 Version 5.0 CP/M Basic	1
Operating Procedure	1
Special Characters	1
Definitions	2
Commands	2
Edit Mode Subcommands	5
Program Statements	6
Print Using Format Field Specifiers	10
Input Output Statements	11
Operators	14
Arithmetic Functions	15
String Functions	17
Input Output Functions	18
Table of Error Codes	19
Standard Keyboard Character Additions	20
Basic-80 Index	21
HARDWARE	
Memory Map	1
Simplified Memory Map	2
Applesoft Variable Maps	3
Applesoft Zero Page Usage	4
Tables 1 thru 17	7 - 12
Figures 1 thru 5	12 - 15
PRINTERS	
Silentype	1
Initialization Procedures	1
Parameter Table	1
Paper Tiger	2
Starting to Use Your New Printer	2
Using Printer Commands in Basic Programs	4
Firmware Description	5
Operating Hints	6
Using the Printer Card as a General-Purpose Parallel Output Port	6

Paper Tiger 440G	7
Paper Tiger 460G	8
Spinwriter Program Cards	9
HARDWARE CARDS	
Special Roms Available For Mountain Hardware ROM + Board.	1
Disk Copy / Disk Space in ROM	1
Explanation of Prompt Questions	2
Catalog CMND Menu / Disk Map in ROM	3
Applesoft Renumber / Merge in ROM	5
'BASICS' in ROM	5
List CTRL in ROM	6
REFERENCE	
Applesoft	1
List	3
DOS	4
Mini-Assembler	6
Monitor ROM	8
Cassette Interface	8
Software Entry	8
Saving the Program to Disk	9
Using the Printer	9
Setting the Print Modes	10
Serial Card	10
Communications Card	11
Hires Shape Loader	12
Hires Shape Saver	12
Renumber Patches	13
Miscellaneous	14
Calls - Peeks, Pokes & Colors	15
Graphics Mode Controls	15
Text Mode Controls	15
Colors as Set by Color	17
ASCII Character Codes	18
Table of the ASCII Codes	19
Octal & Hex to Decimal Conversion	20
PROGRAMS	
Registration Card	

System Monitor

System Monitor Commands

Apple II contains powerful machine level monitor for use by the advanced programmer. To enter the monitor either press RESET button on keyboard or CALL -151 (Hex FF65) from Basic. Apple II will respond with an "*" (asterisk) prompt character on the TV display. This action will not kill current BASIC program which may be re-entered by a C^C (control C). NOTE: "adrs" is a four digit hexadecimal number. Remember to press "return" button at the end of each line.

Command Format

Example

Description

Examine Memory

adrs	*C0F2	Examines (displays) single memory location of (adrs).
adrs1.adrs2	*1024.1048	Examines (displays) range of memory from (adrs1) thru (adrs2).
(return)	*(return)	Examines (displays) next 8 memory locations.
.adrs2	*.4096	Examines (displays) memory from current location thru location (adrs2).

Change Memory

adrs:data data data	*A256:EF 20 43	Deposits data into memory starting at location (adrs).
:data data	*:F0 A2 12	Deposits data into memory starting after (adrs) last used for deposits.

Move Memory

adrs1 adrs2. adrs3M	*100 B010.B410M	Copy the data now in the memory range from (adrs2) to (adrs3) into memory locations starting at (adrs1).
	*800:801 800.8FEM	Fills from 800 to 8FF with 0's

Note 1: 1 less than last location to be filled. Note 2: Fill byte is hex number immediately after colon (:).

Verify Memory

adrs1	adrs2.	*100 B010.B410V	Verify that block of data in memory range from (adrs2) to (adrs3) exactly matches data block starting at memory location (adrs1) and displays differences if any.
	adrs3V		

Cassette I/O

adrs1,adrs2R	*300.4FFR	Reads cassette data into specified memory (adrs) range. Record length must be same as memory range or an error will occur.
adrs1,adrs2W	*800.9FFW	Writes onto cassette, data from specified memory (adrs) range.

Display

I	*I	Set inverse video mode. (Black characters on white background).
N	*N	Set normal video mode. (White characters on black background).

Dis-assembler

adrsL	*C800L	Decodes 20 instructions starting at memory (adrs) into 6502 assembly mnemonic code.
L	*L	Decodes next 20 instructions starting at current memory address.

Mini-assembler

(Turn-on)	*F666G	Turns-on mini-assembler. Prompt character is now a "!" (exclamation point).
\$ (monitor command)	!\$C800L	Executes any monitor command from mini-assembler. Note that many monitor commands change current memory address reference so that it is a good practice to retype desired address reference upon return to mini-assembler.
adrs: (6502 MNEMONIC instruction)	!C010:STA 23FF	Assembles a mnemonic 6502 instruction into machine codes. If error machine will refuse instruction, sound bell, and reprint line with up arrow under error.

(space) (6502 ! STA 01 FF
mnemonic
instruction)

Assembles instruction into next available memory location. (Note space between "!" and instruction)

(TURN-OFF) ! (Reset Button)

Exits mini-assembler and returns to system monitor.

Monitor Program Execution and Debugging

adrsG *300G

Runs machine level program starting at memory (adrs)

adrsT *800T

Traces a program starting at memory location (adrs) and continues trace until hitting a breakpoint. Break occurs on instruction 00 (BRK), and returns control to system monitor. Opens 6502 status registers (see Note 1).

adrsS *C050S

Single steps through program beginning at memory location (adrs). Type a letter S for each additional step that you want displayed. Opens 6502 status registers (see Note 1).

(Control E) *E^C

Displays 6502 status registers and opens them for modification (see Note 1).

(Control Y)* Y^C

Executes user specified machine language subroutine starting at memory location (\$3F8).

Note 1:

6502 status registers are open if they are last line displayed on screen. To change them type ":" then "data" for each register.

Example: A=3C X=FF Y=00 P=32 S=F2

*:FF

Changes A register only

*:FF 00 33

Changes A, X, and Y registers

To change S register, you must first retype data for A, X, Y and P.

Hexidecimal Arithmetic

data1 + data2 *78 + 34

Performs hexidecimal sum of data1 plus data2.

data1 - data2 *AE - 34

Performs hexidecimal difference of data1 minus data2.

Set Input/Output Ports

(X) (Control P) *5P^C

Sets printer output to I/O slot number (X). (see Note 2 below)

(X) (Control K) *2K^C

Sets keyboard input to I/O slot number (X). (see Note 2 below)

Note 2:

Only slots 1 through 7 are addressable in this mode. Address 0 (Ex: 0P^C or 0K^C) resets ports to internal video display and keyboard. These commands will not work unless Apple II interfaces are plugged into specified I/O slot.

Multiple Commands

*100L 400G AFFT

Multiple monitor commands may be given on same line if separated by a "space".

*LLLL

Single letter commands may be repeated without spaces.

Special Control and Editing Characters

"Control" characters are indicated by a super-scripted "C" such as G^C. They are obtained by holding down the CTRL key while typing the specified letter. Control characters are NOT displayed on the TV screen. B^C and C^C must be followed by a carriage return. Screen editing characters are indicated by a super-scripted "E" such as D^E. They are obtained by pressing and releasing the ESC key then typing specified letter. Edit characters send information only to display screen and does not send data to memory. For example, V^C moves cursor to right and copies text while A^E moves cursor to right but does not copy text.

Character

RESET Key

Immediately intercepts any program execution and resets computer. Also sets all text mode with scrolling window at maximum. Control is transferred to System Monitor and Apple prompts with a "*" (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.

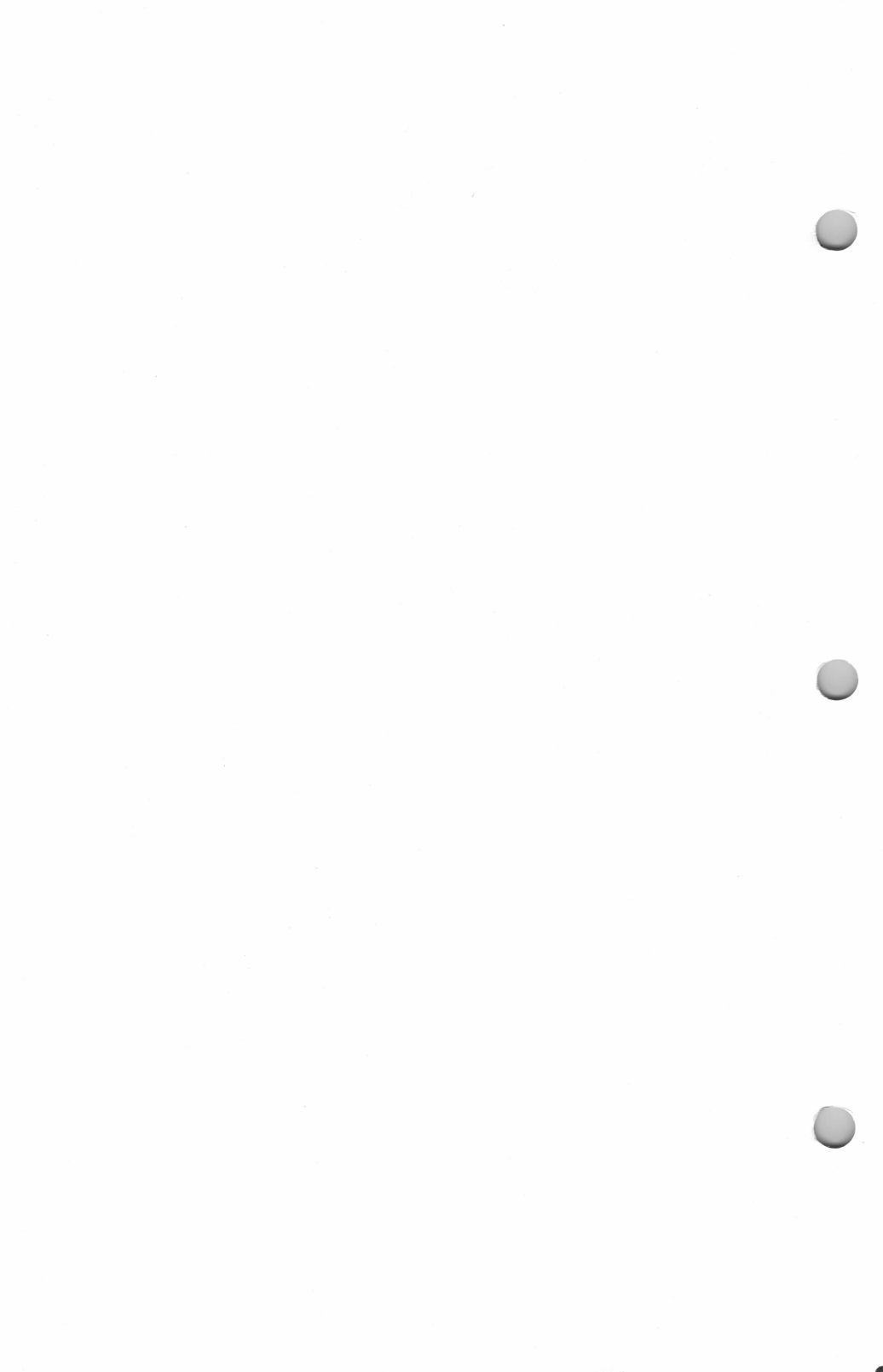
Control B

If in System Monitor (as indicated by a "*"), a control B and a carriage return will transfer control to BASIC, scratching (killing any existing BASIC program) and set HIMEM: to maximum installed user memory and LOMEM: to 208.

Control C	If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in System Monitor, (as indicated by "*"), control C and a carriage return will enter BASIC without killing current program.
Control G	Sounds bell (beeps speaker)
Control H	Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apple supplied keyboards have special key " " on right side of keyboard that provides this function without using control button.
Control J	Issues line feed only.
Control V	Compliment to H ^C . Forward spaces cursor and copies over written characters. Apple keyboards have " → " key on right side which also performs this function.
Control X	Immediately deletes current line.
ESC A	Move cursor to right.
ESC B	Move cursor to left.
ESC C	Move cursor down.
ESC D	Move cursor up.
ESC E	Clear text from cursor to end of line.
ESC F	Clear text from cursor to end of page.
ESC @	Home cursor to top of page, clear text to end of page.

*If BASIC program is expecting keyboard input, you will have to hit carriage return key after typing control C.

NOTES:



DOS

DOS

DISK OPERATING SYSTEM

BOOTING DOS

(S = Slot Containing Disk Drive with DOS)

FROM INTEGER	FROM APPLESOFT	FROM MONITOR
TYPE:		
PR#s	PR#s	Cs00G
IN#s	IN#s	S CTRL-P

INITIALIZING SLAVE DISKETTES

- Boot DOS from system master diskette
- Load or write greeting program (HELLO)
- Remove diskette system master
- Insert new blank diskette or diskette to be erased
- Type INIT HELLO

NOTE: This initialized diskette will only boot on the same memory size system or larger.

CHANGING SLAVE DISKETTES TO MASTER DISKETTES

- Initializing slave diskettes as above must be done
- Insert system master
- Type BRUN MASTER CREATE
- Remove system master diskette
- Answer question for name of greeting program
- Insert slave diskette to be converted to master diskette
- Press [RETURN]

Your slave diskette is now independent of memory size and can be booted on any size memory.

LISTING FILES ON DISKETTE

- Type CATALOG

All files will be shown by listing the file type, number of sectors (256 bytes per sector), and the program name (30 characters maximum)

- File Type
 - A — Applesoft Program
 - I — Interger Program
 - B — Machine Language Program
 - T — Text File

If the file type is preceded by an asterick * then the file is 'locked' (protected from erasure or overwriting)

- File Name
 - Maximum of 30 characters
 - Must start with an alpha character
 - Can contain any character except a comma (,).

DISK COPY PROGRAMS

NOTE: Write Protect Disk To Be Copied

APPLE COPY PROGRAMS

DOS 3.2 — Allows copy only between drives in the same slot. You can copy in either direction.

NOTE: Works only in integer basic

DOS 3.3 — Allows copy from any disk to any disk in any slot including the same disk drive (single disk copy)

NOTE: Works from integer 'copy' or Applesoft 'copy A'

COPY FROM DOS 3.2 to DOS 3.3

Apple provides a program called MUFFIN to transfer selected or all disk files from DOS 3.2 to DOS 3.3

CAUTION — NOTE: Some programs which run in DOS 3.2 may not run in DOS 3.3 so be careful to check their operation before assuming they will run.

- With systems master in disk drive type — BRUN MUFFIN
- Select option 1 'Convert Files'
- Put formatted DOS 3.3 disk in destination drive and DOS 3.2 disk in source drive and answer drive location questions
- For single file type the desired filename. For all or selected files type an equals sign "=".

You will be asked: Do you want prompting?

If you want all files then answer 'NO'

If you want to determine which files are transferred then answer 'YES' and you will be asked to indicate whether or not to convert (copy) that file by typing Y or N or Q to stop and return to the menu.

FID — COPY SINGLE FILES DOS 3.3

(or DOS 3.2 if transferred by Super Disk Copy III)

This is used to copy single files or all files to an initialized diskette.

If during file transfer to the destination disk, the file is found to already exist you are given the option to rename the file being transferred to operate FID

- BRUN FID
- Select Option

For copy routines see the MUFFIN program description.

OTHER COPY PROGRAMS

SUPER DISK COPY III

For copying between DOS 3.2 and DOS 3.3

A program for copying between DOS 3.3 and 3.2 and vice versa is available for \$30.00 from

Sensible Software
6619 Perham Drive
West Bloomfield, Michigan 48033

DISK COPY / DISK SPACE IN ROM

Copies diskettes from either single or dual drive, single or dual controller, 13 or 16 sector and with or without VTOC.

Available from:

Soft CTRL Systems
Box 599
West Milfor, N.J. 07480

RUNNING 13 SECTOR DISKS IN 16 SECTOR SYSTEM

Apple provided software

- Boot BASICS diskette furnished with DOS 3.3 and then insert 3.2 (13 sector) diskette
- or
- BRUN BOOT 13 from system master for DOS 3.3 then insert 13 sector diskette

DISK COMMANDS

[] indicated not needed, if default values are OK.

See abbreviations and prefixed for
definition of single letter abbreviations.

Initialize

INIT f[,v[,Ss][,Dd]

Initializes diskette for system of same memory size only - will not boot on system of larger memory size.

General Program Control

LOAD f [,Ss][,Dd][,Vv]

Loads program of name f from disk.

SAVE f [,Ss][,Dd][,Vv]

Saves program of name f to disk.

DELETE f [,Ss][,Dd][,Vv]

Deletes program of name f from disk.

RENAME f,g[,Ss][,Dd][,Vv]

Renames program f to name g

CATALOG [,Ss][,Dd]

Displays the volume number of diskette and list of all files and file types. File types are:

I - Integer BASIC program

A - Applesoft BASIC program

T - Text file - written by WRITE or APPEND commands

B - Bit-for-bit image of memory locations (machine code)

VERIFY f[,Ss][,Dd][,Vv]

Performs check on disk data to make sure it is self-consistent — does not check disk against memory contents.

RUN f[,Ss][,Dd][,Vv]

Loads and runs program of name f.

CHAIN f [,Ss][,Dd][,Vv]

For Integer Basic Programs only — loads and runs program f, but does not clear variables from previous program.

EXEC f[,Rr][,Ss][,Dd][,Vv]

Command file for program execution as if entered from keyboard.

LOCK f[,Ss][,Dd][,Vv]

When executed on file f it protects the file from deletion or erasure (write protected).

UNLOCK f[,Ss][,Dd][,Vv]

Removes write protect from file f.

Machine Code

BSAVE f,Aa,Lj [,Ss][,Dd][,Vv]

Saves machine code starting at address a for length j.

BLOAD f[,Aa][,Ss][,Dd][,Vv]

Loads machine code.

BRUN f[,Aa][,Ss][,Dd][,Vv]

Loads and runs machine code.

MAXI FILES n

OPEN f[,Lj][,Ss][,Dd][,Vv]

CLOSE f[,Ss][,Dd][,Vv]

READ f[,Rr][,Bb][,Ss][,Dd][,Vv]

WRITE F[,Rr][,Bb][,Ss][,Dd][,Vv]

POSITION f[,Rr][,Ss][,Dd][,Vv]

APPEND f[,Ss][,Dd][,Vv]

NOMON [C][,I][,O]

C - disk commands

I - information to disk

O - output from disk

MON [C][,I][,O]

CTRL/D

FP [,Ss][,Dd][,Vv]

INT

CALL -35

CALL -151

POKE 72, Ø

3DØG

3D3G

CALL 976

Files

Specific number of files that are active at one time (1 to 16)

Allocates 600 byte buffer and prepares to write to beginning of file.

Deallocates buffer for file f or all files if f is not specified and closes file indicated or if unspecified, all files.

Causes following input and get statements to obtain data from file f rather than keyboard.

Causes following print statements to write to file f.

Positions file pointer to indicated line and byte (relative, not absolute).

Opens file f and writes data after last data previously written.

Prevents writing information specified from being displayed on video screen.

Enables commands previously disabled with NOMON.

Must be used to turn off read command

Control

Activates Applesoft ROM or loads Applesoft from disk.

Transfers control to Integer Basic.

Go to monitor.

Go to monitor.

Go to monitor.

Go to Basic and preserve program.

Returns to Basic losing Basic Program and Applesoft if not in ROM.

Activates DOS if in memory.

NOTE: After using DOS commands in monitor type *48:Ø to insure monitor will work correctly.

ROM/RAM APPLESOFT

To change programs

RAM to ROM (to use APPLESOFT ROM Card)

LOAD f

CALL 54514

SAVE f

ROM to RAM (no APPLESOFT ROM Card)

LOAD f

CALL 3314

SAVE f

Abbreviations

f	File Name 1
g	— File Name 2
s	— Slot 0 to 7
v	— Volume 0 to 254
d	— Drive 1 or 2
r	— Record # 0 to 32767
b	— Byte # 0 to 32767
j	— Length Specifier 1 to 32767
n	— # of Files 1 to 16
a	— Starting Address Decimal \$ precedes Hex Numbers
FP	— Applesoft
INT	— Integer Basic

Prefixes

A	— Address Start for Machine Code (decimal)
A\$	— Address Start for Machine Code (Hexadecimal)
I	— Record Length (Decimal) Bytes
L\$	— Record Length (Hexadecimal) Bytes
V	— Volume
S	— Slot (Default last booted)
D	— Drive (Default last booted)
R	— Record Number (Default 0)
B	— Byte # (After r-th CARRIAGE RETURN)
CTRL D	— Precedes any disk function written in a program
[]	— Signifies Optional Specifiers with Prefixes

MEMORY MAPS

MEMORY AREAS OVER-WRITTEN WHEN BOOTING DOS

Location on
any system:

Location on a
48K system:

Highest RAM
memory address →

DOS moves
HIMEM here
[Note 2]

16383 (\$3FFF) →

7424(\$1D00) →

6912 (\$1B00) →

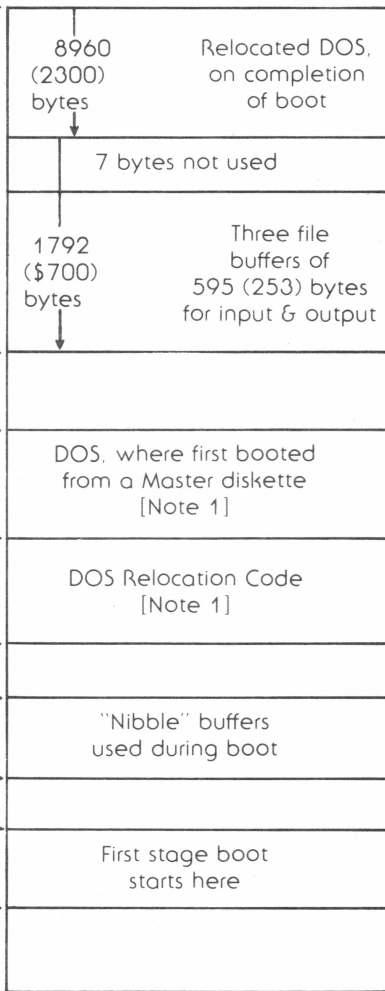
2559 (\$9FF) →

2048 (\$800) →

1023 (\$3FF) →

768 (\$300) →

Lowest RAM
Memory address →
000 (\$000)



← 49151 (\$BFFF)

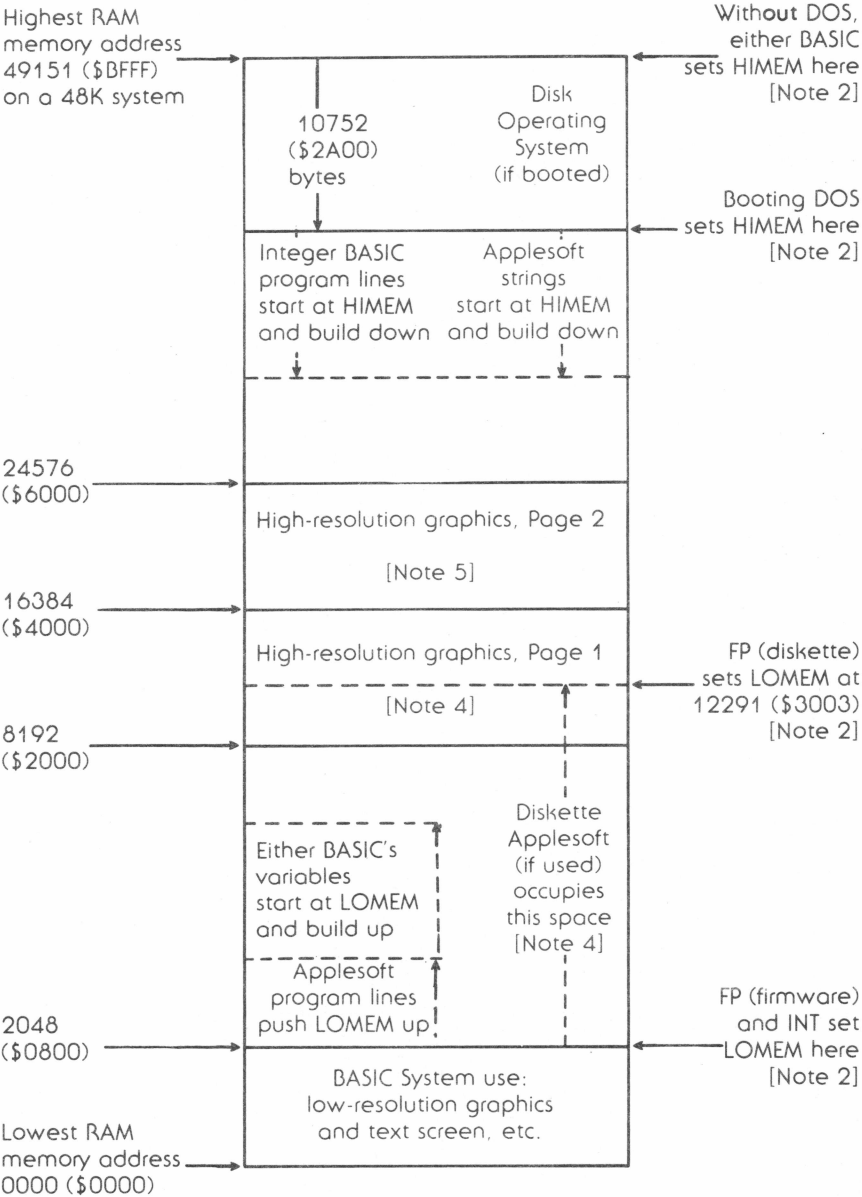
← 40192 (\$9D00)

← 40184 (\$9CF8)

← 38400 (\$9600)

Note 1. This memory area is not affected when booting a **Slave** diskette: DOS is placed directly below the Highest RAM Memory address that was available on the system that **INITIALIZED** the **Slave** diskette, whether appropriate to the present system or not.

MEMORY AREAS USED BY DOS AND EITHER BASIC



Note 2. If your system is in Integer BASIC, the HIMEM pointer can be found (low byte first, then high byte) in locations 76-77 (\$4C-\$4D). If your system is in APPLESOFT BASIC, the HIMEM pointer is in locations 115-116 (\$73-\$74), same format. See Table 2 for the value of HIMEM set by booting DOS. Increasing MAXFILES will move HIMEM down an additional 595 bytes for each file buffer added. For the locations of other Applesoft program pointers, consult your Applesoft II BASIC Programming Manual, Appendix I.

MEMORY AREAS OVER-WRITTEN WHEN BOOTING DOS

Location on
any system:

Location on a
48K system:

Highest RAM
memory address →

DOS moves
HIMEM here
[Note 2] →

16383 (\$3FFF) →

7424 (1D00) →

6912 (\$1B00) →

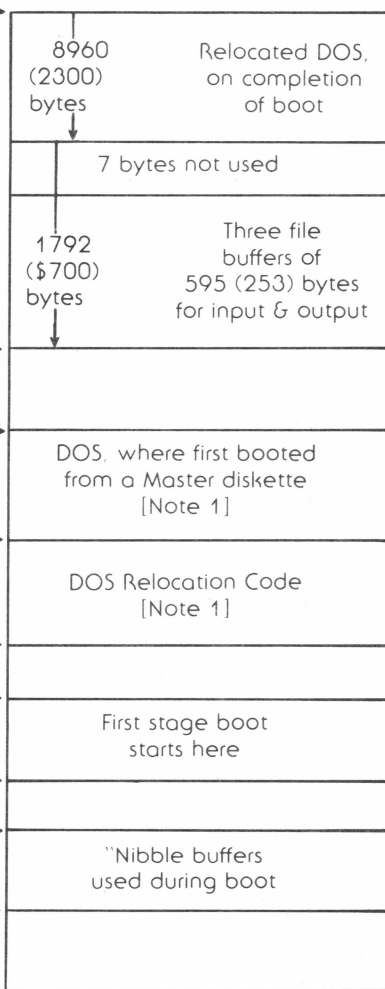
2303 (\$8FF) →

2048 (\$800) →

1023 (\$3FF) →

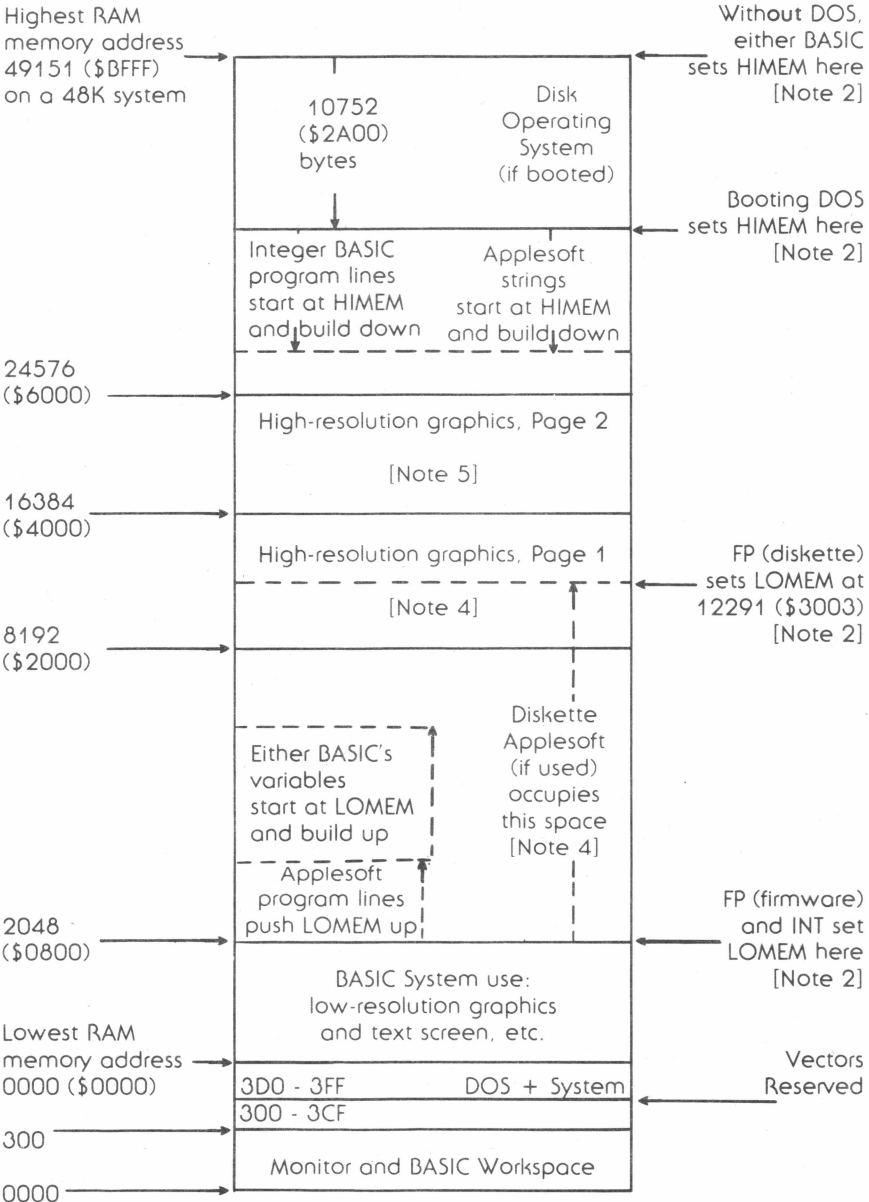
512 (\$200) →

Lowest RAM
Memory address
000 (\$000) →



Note 1. This memory area is not affected when booting a **Slave** diskette: DOS is placed directly below the Highest RAM Memory address that was available on the system that **INITIALIZED** the **Slave** diskette, whether appropriate to the present system or not.

MEMORY AREAS USED BY DOS AND EITHER BASIC



Note 2. If your system is in Integer BASIC, the HIMEM pointer can be found (low byte first, then high byte) in locations 76-77 (\$4C-\$4D). If your system is in APPLESOFT BASIC, the HIMEM pointer is in locations 115-116 (\$73-\$74), same format. See Table 2 for the value of HIMEM set by booting DOS. Increasing MAXFILES will move HIMEM down an additional 595 bytes for each file buffer added. For the locations of other Applesoft program pointers, consult your Applesoft II BASIC Programming Manual, Appendix I.

HIMEM VALUE SET BY BOOTING DOS

When DOS is booted, HIMEM is set according to the amount of memory in the system:

System size	Highest Decimal	RAM address Hexadecimal	HIMEM: Set by DOS boot	
			Decimal	Hexadecimal
16K	16383	\$3FFF	5632	\$1600
20K	20479	\$4FFF	9728	\$2600
24K	24575	\$5FFF	13824	\$3600
32K	32767	\$7FFF	22016	\$5600
36K	36863	\$8FFF	26112	\$6600
48K	49151	\$BFFF	-27136	\$9600 [Note 3]

Note 3. The number -27136 could also be written 38400, but Integer BASIC will not accept numbers greater than 32767. In Integer BASIC, memory addresses greater than 32767 must be expressed as their negative equivalents. The negative equivalent of any positive decimal address n is $(n - 65536)$.

Note 4. Using high-resolution graphics Page 1 erases the contents of memory locations 8192 through 16383. Unless DOS sets HIMEM to a value greater than 16383, an attempt to use high-resolution graphics Page 1 will erase part of DOS. This means that you cannot use Disk II and high-resolution graphics at the same time, unless your system contains at least 32K of memory.

If you are using diskette Applesoft, an attempt to use high-resolution graphics Page 1 will erase part of Applesoft. With diskette Applesoft, you may use high-resolution graphics Page 2, only, if your system contains at least 36K of memory. See Note 5.

Note 5. Using high-resolution graphics Page 2 erases the contents of memory locations 16384 through 24575. Unless DOS sets HIMEM to a value greater than 24575, an attempt to use high-resolution graphics Page 2 may erase part of DOS. This means that you cannot use Disk II and Page 2 high-resolution graphics at the same time, unless your system contains at least 36K of memory.

DOS ENTRY POINTS

Routine to re-connect DOS (if page 3 is over-written):

System size	Decimal address	Hexadecimal address
	(CALL)	(G)
48K	-25153	\$9KDBF
32K	23999	\$5DBF
16K	7615	\$1DBF

The Monitor command 3D0L displays this number at the top right.

Locations containing the start address and length of a BLOADED program:

System size	Start address (low byte)		Program length (low byte)	
	Decimal	Hexadecimal	Decimal	Hexadecimal
48K	43634	\$AA72	43616	AA60
32K	27250	\$6A72	27232	\$6A60
16K	10866	\$2A72	10848	\$2A60

To see the starting address or length after a BLOAD, type PRINT PEEK
(low byte) + PEEK(low byte + 1)*256

ASSEMBLER/
EDITOR

ASSEMBLER / EDITOR

APPLE 6502 ASSEMBLER/EDITOR

FILE NAME: EDASM or INTEDASM or EDASM.OBJ

EDASM — Is an Applesoft editor entry program with date and serial number entry.

INTEDASM — Is an Integer basic editor entry program with date and serial number entry.

Both the above load the machine language file EDASM.OBJ which is the host for the ASSM and the EDITOR.

EDASM.OBJ goes directly to loading the editro program without asking for date or serial number. This program loads the editor first.

COMMAND SYNTAX

If a ? is typed by itself on a line, the help command will display a special command syntax table as shown below:

{ } indicate those characters that do not have to be entered for the command to operate. In the help command the letters that are indicated by { } are actually shown in reverse video.

ASM SOURCE FILE, <OBJECT FILE>,{D}< >,{D}<>

APPEND <LINE> SOURCE FILE

A{DD} <LINE>

CAT{ALOG}

CO{PY} LINE — <LINE> TO LINE

C{HANGE} <BEGIN <—END>> .OLDSTR.NEWSTR.

DR{IVE}

D{ELETE} <BEGIN <—END>>

END

E{DIT} <BEGIN <—END>> .STRING.

FILE

F{IND} <BEGIN <—END>> .STRING.

HI{MEM} =

I{NSERT} LINE

LEN{GTH}

LO{MEM} =

LO{A}D SOURCE FILE

MON

NEW

PR ,<DEVCTL>

P{RINT} <BEGIN <—END>>

R{EPLACE} <BEGIN <—END>>

SL{OT}

S{A}VE <BEGIN <—END>> <SOURCE FILE>

TR{UNC}OF{F}

TR{UNC}ON


```
TLOAD <BEGIN <-END>>
TSAVE <BEGIN <-END>>
T{ABS} <BEGIN <-END>> .END
W{HERE} LINE
```

SYNTAX OF PARAMETER LISTS

Parameter name	Description & Example
Rnumber	Required Number Whole positive integers
Onumber	Optional Number Whole positive integers
Rlinenum	Required decimal line No. Whole positive integers
Olinenum	Optional decimal line No. Whole positive integers
Range	Optional line nol range consisting of one or two line numbers.
Olinenum — single line range	
Olinenum — Olinenum line, no. range	
Olinenum — Start no. to end file	
— Olinenum Start of file to line number.	

If the second line number is smaller than the first, it is taken to mean the total count of the number of lines in the range: example: 100-99 is the same as 100-199.

Rangelist	This is a set of independent ranges, separated by commas. Each range is processed independently of the others, so the line no. ranges can be in any order.
Dstring	A delimited string consists of a delimiter, zero or more alphanumeric characters, and the same delimiter. example: The Period (.) is the delimiter shown below. .EXAMPLE.
Chgstring	A delimited string as above except for a change to a new string. example: The period (.) is the delimiter shown below. .old string.new string.
Rfilename	A required file name 1 to 0 alphanumeric characters not including the comma, naming a DOS TEXT file.
Ofilename	Optional file name 0 to 0 alphanumeric characters not including the comma, naming a DOS TEXT file.

OObfilename	An Optional filename is the same as an Ofilename except that it specifies the name of a Binary or Relocatable object file generated by the assembler.
DevCtrlstring	A string of characters designed to initialize an APPLE peripheral card for use as the assembler's output device.

EDITING COMMANDS

ADD

A{DD} [Olinenum]

If the Olinenum is not specified then new lines will be added at the end of the current data entered into the file.

COPY

CO{PY} Rlinenum1 [—Olinenum] TO Rlinenum3

This command will copy one or more lines TO just before Linenum3 in the file.

CHANGE

C{hange} [Rangelist] Chgstring

If the Rangelist is not specified then all occurrences of the old string will be searched and substituted by the new string.

If the Rangelist is specified then only the lines specified will be searched.

DELETE

D{elete} [Rangelist]

Deletes the lines specified by the range list. A range list is required or an error occurs.

EDIT

E{dit} [Rangelist] [Dstring]

Edits the specified line

To Move Cursor	— use left & right arrows
Replace	— Type over character
Insert	— Type CTRL-I
Insert Off	— Type CTRL-V
Delete	— Type CTRL-D
Restore (orig)	— Type CTRL-R
Find Char.	— Type CTRL-F then Char.
Cancel edit	— Type CTRL-X
Truncate (chop)	— Type CTRL-T
Accept Line	— RETURN

FIND

F{ind} [Rangelist] [Dstring]

Locates and lists all lines containing an occurrence of the Dstring within the lines of the Rangelist. If no Rangelist is specified then the entire file is searched.

INSERT

I{nsert} Rlinenum

Enters input mode, and inserts all the new lines before the given line numbers. CTRL-D or CTRL-Q terminate this mode.

LIST

L{ist} [Rangelist]

Lists the lines of the Rangelist on the screen, showing the current editor line number. If no rangelist is given then the entire file is listed. Listing is halted with a CTRL-C. To suspend the listing use the space bar.

PRINT

P{rint} [Rangelist]

Same as List except that editor line numbers are not shown and the first character is in column 1 of the screen

REPLACE

R{eplace} Range

Deletes the first range and then puts you into Input mode, Inserting at the first line number of the range.

DISK AND TAPE COMMANDS

LOAD

LOAD Rfilename

Reads a file in from the disk drive specified by the current SSlot and DDrive settings.

SAVE

SAVE [Rangelist] [Ofilename]

SAVES all or part of the edit file to the diskette in the current SSlot and DDrive.

APPEND

APPEND [Olinenum] Rfilename

Normally used to load a specified file at the end of the current edit file.

TLOAD

TLOAD [Olinenum]

Loads file from tape. When the Optional line number is specified, TLOAD will insert the tape text before the given line.

TSAVE

TSAVE [Rangelist]

If not Rangelist is used, it writes the entire file to tape in the Assembler/Editor format.

- 10-second leader
- 16-bit file length
- 10-second header
- File data

OPERATING COMMANDS

SLOT

SLOT Rnumber

Sets the SLOT of the disk controller card on which all disk operations will occur.

DRIVE

DRive 1 or DRive 2

Set the particular drive in that slot

CATALOG

CATalog

Lists the Catalog of the disk specified by SLOT and DRIVE.

FILE

FILE

Displays the current SAVE filename along with the DRIVE and SLOT used, the total length of the file and the total bytes remaining in active memory.

HIMEM =

HImem = Rnumber

Sets the highest available address (HIMEM) for the edit file in memory.

LOMEM =

LOmem = Rnumber

Sets the lowest (LOMEM) of the edit file.

LENGTH

LENgth

Displays the current length of the file and the total bytes remaining in unused memory.

MON

MON

Provides easy access to the monitor and return to the editor. To return from the Monitor use CTRL-Y or 3DOG command.

NEW

NEW

Erases the current Editor file.

PR

PR p [,DevCtlstrg]

Sends assembly listings to another device as specified by that Slot.

TRUNCATE

TRuncOn or TRuncOff

Provides a means or TRuncating the displayed lines of the edit file. The purpose of this mode is to display 40 character lines and disregard the rest of the line. There is no effect on the contents of the file, only the display.

TABS

Tabs [TABLIST] [Dstring]

Lets you use the space bar like the tab key of a typewriter to format assembly-language statements as you type them in.

WHERE

Where Rlinenum

This is a tool for manipulating the edit file from the Apple Monitor. It provides a HEX address of the beginning of any line in the file.

END

END

Used to end the Assembler/Editor and return to the Basic language. You can recover by typing CALL 3075. This only works from integer basic as the file will be erased if coming from Applesoft.

EDITOR

For source program entry, tab space is used as a tab. Ther are four fields:

- The Label
- The Operation Code (opcode)
- The Operand
- The Comment

These commands are separated by the spaces and the editor tab function will format the display.

LABEL	—	● 1 to 250 Characters (16 practical limit) ● Must always begin in the first character position of a line ● This is optional field
OPCODE	—	Operation Code 2 or More Letters (Mostek technology mnemonics) Must be Preceded by Space

BASICS

APPLESOFT EXTENDED, FLOATING — POINT BASIC

SIMPLE VARIABLES

Type	Name	Range
Real	AB	+ / - 9.999999999 E+37
Integer	AB%	+ / - 32767
String	AB\$	Ø to 255 characters

Where A is a letter, B is a letter or digit. Name may be more than two characters, but only first two are significant: AB% and AB3Q5% are the same integer variable.

ARRAY VARIABLES

Type	Name of Typical Element
Real	AB(3,12,7)
Integer	AB%(3,12,7)
String	AB\$(3,12,7)

Array size is limited by available memory.

ALGEBRAIC OPERATORS

=	Assigns value to variable (LET optional)
-	Negation
^	Exponentiation
*	Multiplication
/	Division
+	Addition
-	Subtraction

RELATIONAL AND LOGICAL OPERATORS

=	Equal
<>	Not equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
NOT	Logical "Not"
AND	Logical "And"
OR	Logical "Or"

Relational and logical expressions have value 1 if true, Ø if false. Relational operators can also be used to compare strings.

SYSTEM AND UTILITY COMMANDS

LOAD	Loads a program from tape.
LOAD PGM	Loads specified program from disk program name (PGM)
SAVE	Saves a program on tape
SAVE PGM	Saves specified program to disk program name (PGM)
NEW	Clears current program from memory
RUN	Executes program starting at lowest line number.
RUN 100	Executes program starting at line 100
STOP	Halts execution and tells in which line.
END	Halts execution with no message. Used to end program operation.
CTRL-C	Used in immediate mode to halt program or listing.
WITH AUTO START ROM reset	Halts program execution and returns to current operating system Unconditional jump to Monitor, if no Auto Start ROM. Use CTRL-C or ~~ORG~~ to return to APPLESOFT
CONT	Continues program execution stopped by STOP, END or CTRL-C.
TRACE	Debugging aid; lists each line number as it is executed.
NOTRACE	Turns off TRACE.
PEEK (X)	Returns contents of memory location X.
POKE X, 13	Changes contents of memory location X to the value 13.
WAIT X,Y,Z	Waits until contents of location X, when XORed with Z and ANDed with Y, gives non-zero result.
CALL X	Goes to and executes machine-language subroutine beginning at memory location X.
USR (X)	Passes value X to a machine-language subroutine.
HIMEM:	Sets highest memory address available to APPLESOFT program use.
LOMEM:	Sets lowest memory address available to APPLESOFT program use.

EDITING AND FORMAT-RELATED COMMANDS

LIST	Lists entire program.
LIST X-Y	Lists from line X to line Y.
DEL X,Y	Deletes from line X to line Y.
REM TEXT	For writing program comments; (TEXT) ignored by program.
VTAB Y	Moves cursor to line Y (1 to 24)

HTAB X	Moves cursor to position X (1 to 40)
TAB (X)	Only in PRINT statement; moves cursor to position X (1 to 40)
POS (0)	Returns current horizontal position of cursor (0 to 39)
SPC (X)	Only in PRINT statement; puts X spaces between last item printed and next.
HOME	Clears screen and puts cursor at top.
CLEAR	Resets all variables to zero.
FRE (0)	Returns amount of memory still available to user.
FLASH	Sets computer output to flashing on video monitor.
INVERSE	Sets computer output to black on white on video monitor.
NORMAL	Turns off flashing or inverse output.
SPEED=X	Sets character output rate (0 slow to 255 fast).
[ESC] A	Moves cursor one space right.
[ESC] B	Moves cursor one space left.
[ESC] C	Moves cursor one space down.
[ESC] D	Moves cursor one space up.
	Enters character under cursor into memory, and moves cursor one space right.
	Deletes one character from line being typed, and moves cursor one space left.
CTRL-X	Cancel line currently being typed.

ARRAYS AND STRINGS

DIM A(X,Y,Z)	Sets maximum subscripts for A; reserves memory space for $X+1 * Y+1 * Z+1$ real elements, starting with $A(0,0,0)$.
DIM A\$(X,Y)	Sets maximum subscripts for A\$, which may contain $X+1 * Y+1$ strings elements, each of up to 255 characters.
LEN(A\$)	Returns number of characters in A\$
STR\$(X)	Returns numeric value of X, converted to a string.
VAL(A\$)	Returns A\$, up to the first non-numeric character, as a numeric value.
CHR\$(X)	Returns ASCII character whose code is X.
ASC(A\$)	Returns ASCII code for first character of A\$.
LEFT\$(A\$,X)	Returns leftmost X characters of A\$.
RIGHT\$(A\$,X)	Returns rightmost X characters of A\$.
MID\$(A\$,X,Y)	Returns Y characters of A\$, starting at character X.
+	Operator used to concatenate strings.

STORE A	Saves numeric array A on tape. Cannot be used to save string arrays directly.
RECALL B	Loads array back from tape: array B must have been DIMensioned correctly.

INPUT/OUTPUT COMMANDS

(Also see LOAD and SAVE, STORE and RECALL.)

INPUT A\$	Puts ? on screen; waits for user to type a string value for A\$.
INPUT "XYZ";A	Prints XYZ on screen; waits for user to type a real number value for A.
GET A\$	Waits for user to type a one-character value for A\$; does not need RETURN key.
DATA X,"Y",Z	Establishes list of data elements that can be used by READ statements.
READ A\$	Assigns next DATA element to A\$
RESTORE	Starts READING from first DATA element again.
PRINT "X="";X	Prints string X= and value of variable X on screen. Semicolons concatenate printed items, commas separate items into three tab fields. The symbol ? also means PRINT.
IN#6	Takes future input from peripheral device in slot #6, instead of from keyboard (IN#0).
PR#6	Sends output to peripheral device in slot #6, instead of to TV screen (PR#0).
LET X=Y	LET is optional; assigns value of Y to variable X.
DEF FN	Defines a function FNA. In later use, the argument of
A(X)=X+23/X	FNA will be substituted for X in the defined expression. FNA(4) would return 9.75

COMMANDS RELATING TO FLOW OF CONTROL

GOTO 347	Branches to line 347
IF X=3	If the assertion X=3 is true (non-zero), then execution
THEN STOP	continues. If the assertion is false (zero), then execution jumps to the next numbered line.
FOR X=1 TO 0	Executes all statements between the FOR statement
STEP 4 ...	and the corresponding NEXT, first with X=1, then with
NEXT X	X=5, X=9, etc. until X > 20, when execution continues after NEXT. STEP size is 1 if not specified.
NEXT X	Defines bottom of FOR ... NEXT loop. The X is optional.
GOSUB 33	Branches to the subroutine at line 33

RETURN	Marks end of subroutine; returns to statement following most recent GOSUB.
POP	Removes one address from the RETURN address stack.
ON X GOTO	Branches to the Xth line number in the list. If X=2, goes to line 12.
ON X GOSUB	Branches to subroutine at the Xth line number in the list.
ONERR GOTO	Subsequent errors cause branch to error-handling routine at
4500	line 4500 instead of message and program halt.
RESUME	In error-handling routine, causes return to statement where error occurred.

GRAPHICS AND GAME CONTROLS

Low-Resolution Graphics

GR	Sets low-resolution graphics; clears top 40 x 40 area to black; bottom 4 lines text.
COLOR=X	Sets color (0 to 15) for next plotting.
PLOT X,Y	Places colored dot at horizontal coordinate X and vertical coordinate Y. X and Y are from 0 to 39. 0,0 is top left.
HLIN X1,X2 AT Y	Draws horizontal line from the point at X2,Y.
VLIN Y1,Y2 AT X	Draws vertical line from the point at X,Y1 to the point at X,Y2.
SCRN(X,Y)	Returns color on screen at the point X,Y.

High-Resolution Graphics

HGR	Sets high-resolution graphics, page 1; clears top 280 x 160 area to black; bottom 4 lines text.
HGR2	Sets high-resolution graphics, page 2; clears entire 280 x 192 screen to black.
HCOLOR=X	Sets color (0 to 7) for next plotting.
HPlot X,Y	Places colored dot at horizontal coordinate X and vertical coordinate Y. X is from 0 to 279; Y is from 0 to 159 (HGR) or to 191 (HGR2). 0,0 is top left corner.
HPlot X1,Y1 TO X2,Y2	Draws line from the point at X1,Y1 to the point at X2,Y2. Command may be extended to additional points ... TO XN,YN.
SHLOAD	Loads a shape table from tape.

DRAW 3 AT X,Y Draws shape definition #3 from a previously loaded shape table, starting at the point X,Y in color set by HCOLOR.

XDRAW 3 AT X,Y Draws shape definition #3 from shape table; color of each point plotted is complement of color on screen at that point.

ROT=X Sets rotation of shape for DRAW or XDRAW. ROT=0 is vertical, ROT=16 is 90 degrees clockwise, ROT=32 is 180 degrees clockwise, etc.

SCALE=X Sets scale (1 to 255) of shape for DRAW or XDRAW.

Game Controls

PDL(X) Returns setting from 0 to 255 of game control X (0 to 3).

PEEK (X-16287) If >127, button on game control X (0 to 2) is being pressed.

PEEK(-16336) "Clicks" APPLE's speaker.

SOME MATH FUNCTIONS

SIN(X) Returns sine of X radians.

COS(X) Returns cosine of X radians.

TAN(X) Returns tangent of X radians.

ATN(X) Returns arctangent, in radians, of X.

INT(X) Returns largest integer less than or equal to X.

RND(1) Returns random real number from 0 to 0.99999999 each time used.

RND(0) Returns last random number again.

RND(-3) Returns 4.48217179E-08. A different fixed number is returned for each different negative argument. After this, RND with positive argument will follow a fixed sequence.

SGN(X) Returns -1 if $X < 0$, 0 if $X = 0$, and 1 if $X > 0$.

ABS(X) Returns absolute value of X.

SQR(X) Returns positive square root of X.

EXP(X) Returns e (2.718289) to the power X.

LOG(X) Returns natural logarithm of X.

For Disk Functions see DOS Section — CTRL-D or CHR\$(4) must be printed prior to those commands to be used from APPLESOFT.

Alphabetic Index to APPLESOFT BASIC Commands

Command	Page	Command	Page	Command	Page
ABS	6	HPlot	5	repeat	
arrow keys	3	HTAB	3	reset	2
ASC	3			RESTORE	4
ATN	6	IF...GOTO		RESUME	5
		IF...THEN		RETURN	5
CALL	2	INPUT	4	RIGHT\$	3
CHR\$	3	INT	6	right arrow	3
CLEAR	3	INVERSE	3	ROT	6
COLOR	5	IN#	4	RND	6
CONT	2			RUN	2
COS	6	LEFT\$	3		
CTRL-C	2	left arrow	3	SAVE	2
CTRL-X	3	LEN	3	SCALE	6
		LET	4	SCRN	5
DATA	4	LIST	2	SGN	6
DEF FN	4	LOAD	2	SHLOAD	5
DEL	2	LOG	6	SIN	6
DIM	3	LOMEM:	2	SPC	3
DRAW	6			SPEED	3
		MID\$	3	SQR	6
END	2			STEP	
ESC-A	3	NEW	2	STOP	2
ESC-B	3	NEXT	4	STORE	4
ESC-C	3	NORMAL	3	STR\$	3
ESC-D	3	NOTRACE	2		
EXP	6			TAB	3
		ON ... GOSUB	5	TAN	6
FOR...TO...STEP		ON ... GOTO	5	TEXT	
FLASH	3	ONNERR GOTO	5	TRACE	2
FRE	3				
		PDL	6	USR	2
GET	4	PEEK	2		
GOSUB	4	PLOT	5	VAL	3
GOTO	4	POKE	2	VLIN	5
GR	5	POP	5	VTAB	2
		POS	3		
HColor	5	PRINT	4	WAIT	2
HGR	5	PR\$	4		
HGR2	5			XDRAW	6
HIMEM:	2	READ	4		
HLIN	5	RECALL	4		
HOME	3	REM	2		

High-Resolution Operating Subroutines

(ROM Location shown in parenthesis)

Other locations are for HIRES software.

Hex	BASIC Example	Description
0C00 (D000)	CALL 3072 (CALL-12288)	INIT — Initializes High-Resolution Graphics mode.

This subroutine sets High-Resolution Graphics mode with a 280 x 160 matrix of dots in the top portion of the screen and four lines of text in the bottom portion of the screen. INIT also clears the screen.

0C0E (D00E)	CALL 3086 (CALL-12274)	CLEAR — Clears the screen.
----------------	---------------------------	----------------------------

This subroutine clears the High-Resolution screen without resetting the High-Resolution Graphics mode.

0C7C (D07C)	CALL 3780 (CALL-11580)	PLOT — Plots a point on the screen.
----------------	---------------------------	-------------------------------------

This subroutine plots a single point on the screen.

	Location	
Y (vertical) Coordinate	802 Decimal A - register	Range 0 Top of Screen 159 Bottom of Screen
X (horizontal) Coordinate	800 Decimal 801 Decimal X(X LO) Register Y(Y HI) Register	Range 0 (left) to 279 (right) Loc 800 = X mod 256 801 = X / 256
Color	812 032C Hex	4 Colors 0 (\$0) Black 85 (\$55) Green 170 (\$AA) Violet 255 (\$FF) White

0C26 (D026)	CALL 3761 (CALL-11599)	POSN — Positions a point on the screen.
----------------	---------------------------	---

This subroutine does all calculations for a PLOT, but does not plot a point (it leaves the screen unchanged). This is useful when used in conjunction with LINE or SHAPE (described later). To use this subroutine set up the X and Y coordinates just the same as for PLOT. The color in location 812 (\$32C) is ignored.

0C95 (D095)	CALL 3786 (CALL-11574)	LINE — Draw a line on the screen.
----------------	---------------------------	-----------------------------------

This subroutine draws a line from the last point PLOTEd or POSN'ed to the other endpoint which is passed in the same manner as for a PLOT or POSN (800, 801, 802). The color of the line is set in location 812 (\$32C). After the line is drawn, the new endpoint becomes the base endpoint for the next line drawn.

0DBC CALL 3805
(D1BC) (CALL-11555)

SHAPE — Draws a predefined shape on the screen.

This subroutine draws a predefined shape on the screen at the point previously PLOTEd or POSN'ed (800, 801, 802). The shape is defined by a table of vectors in memory.

Hex	Decimal	Description
032D	804 (LO)	X mod 256 } Starting Address of X/256 } Shape Table
032E	805 (HI)	
001C	28	Color of Shape (see plot description)
032F	806	Scaling Factor (1 = same size 2 = twice size, etc.)
0330	807	Rotation Factor 0 to 64 (0 = right side up 16 = 90° clockwise).

Decimal Tokens for Keywords

Hex	Decimal Token	Keyword	Hex	Decimal Token	Keyword
80	128	END	95	149	XDRAW
81	129	FOR	96	150	HTAB
82	130	NEXT	97	151	HOME
83	131	DATA	98	152	ROT =
84	132	INPUT	99	153	SCALE =
85	133	DEL	9A	154	SHLOAD
86	134	DIM	9B	155	TRACE
87	135	READ	9C	156	NOTRACE
88	136	GR	9D	157	NORMAL
89	137	TEXT	9E	158	INVERSE
8A	138	PR#	9F	159	FLASH
8B	139	IN#	A0	160	COLOR =
8C	140	CALL	A1	161	POP
8D	141	PLOT	A2	162	VTAB
8E	142	HLIN	A3	163	HIMEN:
8F	143	VLIN	A4	164	LOMEN:
90	144	HGR 2	A5	165	ONERR
91	145	HGR	A6	166	RESUME
92	146	HCOLOR =	A7	167	RECALL
93	147	HPLOT	A8	168	STORE
94	148	DRAW	A9	169	SPEED =

Hex	Decimal Token	Keyword
AA	170	LET
AB	171	GOTO
AC	172	RUN
AD	173	IF
AE	174	RESTORE
AF	175	&
B0	176	GOSUB
B1	177	RETURN
B2	178	REM
B3	179	STOP
B4	180	ON
B5	181	WAIT
B6	182	LOAD
B7	183	SAVE
B8	184	DEF
B9	185	POKE
BA	186	PRINT
BB	187	CONT
BC	188	LIST
BD	189	CLEAR
BE	190	GET
BF	191	NEW
C0	192	TAB(
C1	193	TO
C2	194	FN
C3	195	SPC(
C4	196	THEN
C5	197	AT
C6	198	NOT
C7	199	STEP
C8	200	+
C9	201	—
CA	202	*

Hex	Decimal Token	Keyword
CB	203	/
CC	204	^
CD	205	AND
CE	206	OR
CF	207	>
D0	208	=
D1	209	<
D2	210	SGN
D3	211	INT
D4	212	ABS
D5	213	USR
D6	214	FRE
D7	215	SCRNC
D8	216	PDL
D9	217	POS
DA	218	SQR
DB	219	RND
DC	220	LOG
DD	221	EXP
DE	222	COS
DF	223	SIN
E0	224	TAN
E1	225	ATN
E2	226	PEEK
E3	227	LEN
E4	228	STR\$
E5	229	VAL
E6	230	ASC
E7	231	CHR\$
E8	232	LEFT\$
E9	233	RIGHT\$
EA	234	MID\$

ONERR GOTO CODES

codeDOS message

Most common cause

- | | |
|--------------------------|-------------------------------------|
| 1 LANGUAGE NOT AVAILABLE | Applesoft not on diskette |
| 2,3 RANGE ERROR | Command parameter too large |
| 4 WRITE PROTECTED | Write-protect tab on diskette |
| 5 END OF DATA | READING beyond end of text file |
| 6 FILE NOT FOUND | File misspelled, or not on diskette |
| 7 VOLUME MISMATCH | Wrong Volume parameter |
| 8 I/O ERROR | Door open, or diskette not INITed |
| 9 DISK FULL | Too many files on diskette |

10 FILE LOCKED	Attempt to over-write a LOCKed file
11 SYNTAX ERROR	Bad file name, parameter, or comma
12 NO BUFFERS AVAILABLE	Too many text files OPEN
13 FILE TYPE MISMATCH	Diskette file doesn't match command
14 PROGRAM TOO LARGE	Insufficient Apple memory available
15 NOT DIRECT COMMAND	Command must be in a program

USING TRACE FROM APPLESOFT & ITS AFFECTS WITH DOS

The trace command that can be used within applesoft does not work with disk commands as a line number is printed before the DOS command.

A partial solution that works for DOS commands except if DOS tries to READ an INPUT from the diskette, instead it will stop and wait for input from the keyboard. No current solution to this problem exists. Disk commands preceded by CTRL-D

Normally D\$ = CHR\$(13) CTRL-D

Partial solution

Let D\$ = CHR\$(13) + CHR\$(4)
CTRL-D [RETURN]

Most DOS command except that stated above will work.

Reserved Words in Applesoft & Integer Basic

¹Integer Basic Only

²Applesoft Basic Only

G²

ABS ¹	AND ²	ASC ¹	AT ¹	ATN ¹	AUTO ¹		
CALL ¹	CHR\$ ²	CLEAR ²	CLR ¹	COLOR ¹	CON ¹	CONT ²	COS
DATA ²	DEF ²	DEL ¹	DIM ¹	DRAW ²	DSP ¹		
END ¹	EXP ²						
FLASH ²	FN ²	FOR ¹	FRE ²				
GET ²	GOSUB ¹	GOTO ¹	GR ¹				
HCOLOR ²	HGR ²	HGR2 ²	HIMEM: ¹	HLIN ¹	HOME ²	HPlot ²	HTAB ²
IF ¹	IN# ¹	INPUT ¹	INT ²	INVERSE ²			
LEFT\$ ²	LEN ¹	LET ¹	LIST ¹	LOAD ¹	LOG ¹	LOMEM: ¹	
MID\$ ²	MAN ¹						
NEW ¹	NEXT ¹	NORMAL ²	NOT ²	NOTRACE ¹	NO DSP ¹		
ON ²	ONERR ²	OR ²					
PDL ¹	PEEK ¹	PLOT ¹	POKE ¹	POP ¹	POS ²	PRINT ¹	PR# ¹
READ ²	RECALL ²	REM ¹	RESTORE ²	RESUME ²	RETURN ¹	RIGHT\$ ²	
RND ¹	ROT = ²	RUN ¹					
SAVE ¹	SCALE = ²	SCRN ¹	SGN ¹	SHLOAD ²	SIN ¹	SPC ¹	
	SPEED = ²	SQR ²	STEP ¹	STOP ²	STORE ²	STR\$ ²	
TAB ¹	TAN ²	TEXT ¹	THEN ¹	TO ¹	TRACE ¹		
USR ²							
VAL ²	VLIN ¹	VTAB ¹					
WAIT ²							
XPLOT ²	XDRAW ²						

APPLESOFT "tokenizes" these reserved words: each word takes up only one byte of program storage. All other characters in program storage use up only one byte of program storage each.

NOTE

The ampersand (&) is intended for the computer's internal use only; it is **not** a proper APPLESOFT command. This symbol, when executed as an instruction, causes an unconditional jump to location \$3F5. Use reset ctrl C return to recover.

XPLOT is a reserved word that does not correspond to a current APPLESOFT command.

Reserved words recognized by APPLESOFT only in certain contexts.

COLOR, HCOLOR, SCALE, SPEED, and ROT

parse as reserved words only if the next non-space character is the replacement sign, =. This is of little benefit in the case of COLOR and HCOLOR, as the included reserved word OR prevents their use in variable names anyway.

SCRN, SPC and TAB

parse as reserved words only if the next non-space character is a left parenthesis, (.

HIMEM:

must have its colon (:) to be parsed as a reserved word.

LOMEM:

also requires a colon (:) if it is to be parsed as a reserved word.

ATN

is parsed as reserved word only if there is no space between the T and the N. If a space occurs between the T and the N, the reserved word AT is parsed, instead of ATN.

TO

is parsed as a reserved word **unless** preceded by an A **and** there is a space between the T and the O. If a space occurs between the T and the O, the reserved word AT is parsed instead of TO.

Sometimes parentheses can be used to get around reserved words:

	100 FOR A = LOFT OR CAT TO 15
LISTs as	100 FOR A = LOF TO RC AT TO 15
but	100 FOR A = (LOFT) OR (CAT) TO 15
LISTs as	100 FOR A = (LOFT) OR (C AT) TO 15

APPLESOFT ROM CARD

Applesoft/Integer Movement

[RESET] C080 [RETURN] To initialize APPLESOFT.
CTRL/B [RETURN]

[RESET] C081 [RETURN] To initialize Integer Basic.
CTRL/B [RETURN]

ALGEBRAIC OPERATORS

=	Assigns value to variable (LET optional)
-	Negation
^	Exponentiation
*	Multiplication
/	Division
+	Addition
--	Subtraction

RELATIONAL AND LOGICAL OPERATORS

=	Equal
<>	Not equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
NOT	Logical "Not"
AND	Logical "And"
OR	Logical "Or"

Relational and logical expressions have value 1 if true, 0 is false. Relational operators can also be used to compare strings.

APPLESOFT HIRES ROUTINES

These are the entry points to program machine level high resolution graphics with the routines in ROM APPLESOFT.

ZERO PAGE LOCATIONS

HPAG	\$E6	\$20 = PAGE 1, \$40 = PAGE 2
ROT	\$F9	USED WITH SHAPE ROUTINES
SCALE	\$E7	USED WITH SHAPE ROUTINES

THE GRAPHICS ROUTINES

HGR2	\$F3DE	INITIALIZE TO PAGE 2
HGR	\$F3E2	INITIALIZE TO PAGE 1
HCLR	\$F3F2	CLEAR CURRENT SCREEN TO BLACK
BKGND	\$F3F2	CLEAR CURRENT SCREEN TO LAST PLOTTED HCOLOR
HPOSN	\$F411	POSITIONS HIRES CURSOR WITHOUT PLOTTING. A=Y X=XL Y=XH
HPLOT	\$F457	POSITIONS HIRES CURSOR AND PLOTS A POINT. A=Y X=XL Y=XH
HLIN	\$F530	DRAW A LINE FROM LAST POINT TO A=XL X=XH Y=Y
DRAW	\$F5CB	CONVERT LAST POINT'S POSITION TO X—Y COORDINATES. THE RESULTS ARE IN \$EO=XL \$E1=XH \$E2=Y
XDRAW	\$F65D	SAME AS DRAW EXCEPT DOES AN EXCLUSIVE OR WHEN PLOTS ON THE SCREEN.
SETHCOL	\$F6EC	A=COLOR. MUST BE LESS THAN 8
SHLOAD	\$F775	LOADS SHAPE TABLE ABOVE HIMEM (\$73,74)

APPLESOFT ERROR MESSAGES

ERRMSG — \$D412 (\$C12 RAM)

Prints various error messages. On entry, x register contains offset to appropriate message string. Message strings begin at location \$D260 (\$A60 RAM).

X	Message
\$00	NEXT WITHOUT FOR
\$10	SYNTAX
\$16	RETURN WITHOUT GOSUB
\$2A	OUT OF DATA
\$35	ILLEGAL QUANTITY
\$45	OVERFLOW
\$4D	OUT OF MEMORY
\$5A	UNDEF'D STATEMENT
\$6B	BAD SUBSCRIPT
\$78	REDIM'D ARRAY
\$85	DIVISION BY ZERO
\$95	ILLEGAL DIRECT
\$A3	TYPE MISMATCH
\$B0	STRING TOO LONG
\$BF	FORMULA TOO COMPLEX
\$D2	CAN'T CONTINUE
\$E0	UNDEF'D FUNCTION

Example:

LDX #\$85 ;point to message
JMP \$D412 ;indicate division by zero

ERROR CODES

Error detected if PEEK(216) 127

POKE 216,0 — resets ERRFLG

Y=PEEK(212) — Y is error code returned in decimal

COMMANDS RELATED TO ERRORS

$340 X = \text{PEEK}(218) + \text{PEEK}(219) * 256$

This statement sets X equal to the line number of the statement where an error occurred if an ONERRGOTO statement has been executed.

Hex	Decimal Value	Error Type Encountered
0	0	NEXT without FOR
10	16	Syntax
16	22	RETURN without GOSUB
2A	42	Out of DATA
35	53	Illegal Quantity
45	69	Overflow
4D	77	Out of Memory
5A	90	Undefined Statement

6B	107	Bad Subscript
78	120	Redimensioned Array
85	133	Division by Zero
A3	163	Type Mismatch
B0	176	String Too Long
BF	191	Formula Too Complex
E0	224	Undefined Function
FE	254	Bad Response to an INPUT Statement
FF	255	Ctrl C Interrupt Attempted

POKE 768, 104 : POKE 769, 168 : POKE 770, 104 : POKE 771, 166:
POKE 772, 223 : POKE 773, 154 : POKE 774, 72 : POKE 775, 152:

POKE 776, 72 : POKE 777, 96

Establishes a machine-language subroutine at location 768, which can be used in an error-handling routine. Clears up some ONERR GOTO problems with PRINT and ?OUT OF MEMORY ERROR messages. Use the command CALL 768 in the error-handling routine.

DISK ERROR MESSAGES

Error Code At Decimal 222 (Hex DE)

Hex	Decimal Value	Error Messages
04	4	Write Protect
05	5	End of Data
06	6	File Not Found
07	7	Volume Mismatch
08	8	DISK I/O
09	9	Disk Full
0A	10	File Locked
0B	11	CMD Syntax
0C	12	No File Buffs Available
0D	13	Not Basic Program
0E	14	Program Too Large
0F	15	Not Binary File Error

APPLESOFT FLOATING POINT FORMAT

Variables are stored in memory in the following format:

1st byte — two's complement exponent with bit 7 inverted

2nd byte — most significant of signed magnitude mantissa

bit 7 is sign bit

bit 7 = 0 implies non-negative

bit 7 = 1 implies negative

the mantissa is normalized

an implied "1" is hidden under bit 7

3rd byte — second most significant byte of mantissa

4th byte — third most significant byte of mantissa

5th byte — least significant byte of mantissa

	EXP	MSB	2SB	3SB	LSB
VARIABLE X	X	X + 1	X + 2	X + 3	X + 4
0.	\$00	\$00	\$00	\$00	\$00
1.	\$81	\$00	\$00	\$00	\$00
2.	\$82	\$00	\$00	\$00	\$00
3.	\$82	\$40	\$00	\$00	\$00
4.	\$83	\$00	\$00	\$00	\$00
—1.	\$81	\$80	\$00	\$00	\$00
1/3	\$7F	\$2A	\$AA	\$AA	\$AB
—1/3	\$7F	\$AA	\$AA	\$AA	\$AA

Applesoft's internal floating point accumulators follow a similar format, except that bit 7 of the MSB of the mantissa is not hidden under the sign bit.

	EXP	MSB	2SB	3SB	LSB	SGN	TMP
ACC1	\$9D	\$9E	\$9F	\$A0	\$A1	\$A2	\$A3
ACC2	\$A5	\$A6	\$A7	\$A8	\$A9	\$AA	\$AB
0.	\$00	\$00	\$00	\$00	\$00	\$00	\$00
1.	\$81	\$80	\$00	\$00	\$00	\$00	\$00
2.	\$82	\$80	\$00	\$00	\$00	\$00	\$00
3.	\$82	\$C0	\$00	\$00	\$00	\$00	\$00
4.	\$83	\$80	\$00	\$00	\$00	\$00	\$00
1.5	\$81	\$C0	\$00	\$00	\$00	\$00	\$00
—1.	\$81	\$80	\$00	\$00	\$00	\$FF	\$00

SOME APPLESOFT INTERNAL ROUTINES

<routine> address ROM> (address RAM)

<function>

CHRGET \$00B1 (\$00B1)

returns next character in accumulator

zero flag set if end of line or colon

carry flag set if non-numeric

carry flag clear if numeric

advances character pointer at \$B8, \$B9

CHRGOT \$00B7 (\$00B7)

same as CHRGET, but does not advance pointer at \$B8, \$B9

EVALNUM \$DD67 (\$156A)

evaluates numeric expression

leaves result in floating point accumulator

indicates error if non-numeric expression

EVAL \$DD7B (\$157E)

evaluates numeric or string expression
if numeric, leaves result in floating point accumulator
if string, leaves pointer to string results in \$9E, \$9F

FIX \$E752 (\$1F49)

converts floating point accumulator to 16 bit integer
leaves result in \$50, \$51

ERRMSG \$D412 (\$0C12)

prints various error messages
X register is offset to appropriate message string
message strings begin at \$D260 (\$0A60)

GETADR \$DFE3 (\$17E6)

gets address of variable
returns with low byte of address in A and high byte in Y

GETBYTE \$E6F8 (\$1EEF)

evaluates expression and returns as 8 bit value in X

FLOATY \$E301 (\$1AFA)

floats 8 bit value in Y register
leaves result in floating point accumulator

The best way to use these and other Applesoft internal routines is to find an existing Applesoft command or function which is similar in nature to your desired task. Look up the address of this Applesoft routine using the Applesoft Commands and Functions table, and use the existing Applesoft code as a guideline for your own routine. Example:

LET (existing Applesoft operation)

Syntax is:

LET A = <exp>

where A is assigned the value of expression <exp>.

```
LET:          JSR GETADR
               STA $85
               STY $86
               LDA #' =
               PHA
               LDA $11
               PHA
               JSR EVAL ;replace this step for custom function
ASSIGN:       PLA
```

•
•
•

G hires SCRN function (desired custom function)

Syntax is:

G A = X, Y

where A will be assigned 1 or 0 depending on state of pixel at hires coordinate X,Y.

SCRN:	JSR GETADR
	STA \$85
	STY \$86
	LDA #' =
	JSR CMPCHR
	LDA \$12
	PHA
	LDA \$11
	PHA
	JSR GETXY
	JSR POSITION
	LDA BITMASK
	AND (HBASL),Y
	BEQ %1
	LDA #1
%1:	TAY
	JSR FLOATY
	JMP ASSIGN

Emulating existing routines in this manner will expedite development of your own custom functions.

RE: FALSE I/O READ STROBES

How to avoid false reads when writing to I/O devices:

1. Use absolute addressing.
OK, but not slot independent.
2. Use indexed addressing and cross a page boundary.

This method causes a false read from RAM locations, but not from any I/O devices. A false I/O read (before a write) occurs only when the indexing does not cause a carry into the high byte of the address, which takes one extra cycle to process. Example:

```
LDY #$10
STA $C080,Y
```

causes a false read at \$C090 since \$80 + \$10 produces no carry, while

```
LDY #$91
STA $BFFF,Y
```

still stores the accumulator at \$C090, but causes a false read at \$BF90 — not at \$C090. Therefore, no false I/O read is generated. Use of this "trick" will simplify the design of some I/O devices while maintaining code slot-independence.

APPLESOFT COMMANDS AND FUNCTIONS

COMMAND WORD	TOKEN	ADDRESS OF ASCII STRING		ADDRESS OF POINTER TO SUBROUTINE		ADDRESS OF SUBROUTINE	
		RAM	ROM	RAM	ROM	RAM	ROM
END	\$80	\$8D0	\$D0D0	\$800	\$D000	\$1072	\$D870
FOR	\$81	\$8D3	\$D0D3	\$802	\$D002	\$0F68	\$D766
NEXT	\$82	\$8D6	\$D0D6	\$804	\$D004	\$14FC	\$DC9F
DATA	\$83	\$8DA	\$D0DA	\$806	\$D006	\$1197	\$D995
INPUT	\$84	\$8DE	\$D0DE	\$808	\$D008	\$13B5	\$DBB2
DEL	\$85	\$8E3	\$D0E3	\$80A	\$D00A	\$2B38	\$F331
DIM	\$86	\$8E6	\$D0E6	\$80C	\$D00C	\$17DC	\$DFD9
READ	\$87	\$8E9	\$D0E9	\$80E	\$D00E	\$13E5	\$DBE2
GR	\$88	\$8ED	\$D0ED	\$810	\$D010	\$2B8C	\$F390
TEXT	\$89	\$8EF	\$D0EF	\$812	\$D012	\$2B95	\$F399
PR#	\$8A	\$8F3	\$D0F3	\$814	\$D014	\$29EC	\$F1E5
IN#	\$8B	\$8F6	\$D0F6	\$816	\$D016	\$29E5	\$F1DE
CALL	\$8C	\$8F9	\$D0F9	\$818	\$D018	\$29DC	\$F1D5
PLOT	\$8D	\$8FD	\$D0FD	\$81A	\$D01A	\$2A2C	\$F225
HLIN	\$8E	\$901	\$D101	\$81C	\$D01C	\$2A39	\$F232
VLIN	\$8F	\$905	\$D105	\$81E	\$D01E	\$2A48	\$F241
HGR2	\$90	\$909	\$D109	\$820	\$D020	\$2BD4	\$F3D8
HGR	\$91	\$90D	\$D10D	\$822	\$D022	\$2BDE	\$F3E2
HCOLOR=	\$92	\$910	\$D110	\$824	\$D024	\$2EE5	\$F6E9
HPLOT	\$93	\$917	\$D117	\$826	\$D026	\$2EFA	\$F6FE
DRAW	\$94	\$91C	\$D11C	\$828	\$D028	\$2F62	\$F769
XDRAW	\$95	\$920	\$D120	\$82A	\$D02A	\$2F68	\$F76F
HTAB	\$96	\$925	\$D125	\$82C	\$D02C	\$2FE0	\$F7E7
HOME	\$97	\$929	\$D129	\$82E	\$D02E	\$FC58	\$FC58
ROT=	\$98	\$92D	\$D12D	\$830	\$D030	\$2F1A	\$F721
SCALE=	\$99	\$931	\$D131	\$832	\$D032	\$2F20	\$F727
SHLOAD	\$9A	\$937	\$D137	\$834	\$D034	\$2F6E	\$F775
TRACE	\$9B	\$93D	\$D13D	\$836	\$D036	\$2A74	\$F26D
NOTRACE	\$9C	\$942	\$D142	\$838	\$D038	\$2A76	\$F26F
NORMAL	\$9D	\$949	\$D149	\$83A	\$D03A	\$2A7A	\$F273
INVERSE	\$9E	\$94F	\$D14F	\$83C	\$D03C	\$2A7E	\$F277
FLASH	\$9F	\$956	\$D156	\$83E	\$D03E	\$2A87	\$F280
COLOR=	\$A0	\$95B	\$D15B	\$840	\$D040	\$2A56	\$F24F
POP	\$A1	\$961	\$D161	\$842	\$D042	\$116D	\$D96B
VTAB	\$A2	\$964	\$D164	\$844	\$D044	\$2A5D	\$F256
HIMEM:	\$A3	\$968	\$D168	\$846	\$D046	\$2A8D	\$F286
LOMEM:	\$A4	\$96E	\$D16E	\$848	\$D048	\$2AAD	\$F2A5
ONERR	\$A5	\$974	\$D174	\$84A	\$D04A	\$2AD2	\$F2CB
RESUME	\$A6	\$979	\$D179	\$84C	\$D04C	\$2B1F	\$F318
RECALL	\$A7	\$97F	\$D17F	\$84E	\$D04E	\$2BB8	\$F3BC
STORE	\$A8	\$985	\$D185	\$850	\$D050	\$2B9B	\$F39F
SPEED=	\$A9	\$98A	\$D18A	\$852	\$D052	\$2A69	\$F262
LET	\$AA	\$990	\$D190	\$854	\$D054	\$1248	\$DA46
GOTO	\$AB	\$993	\$D193	\$856	\$D056	\$1140	\$D93E
RUN	\$AC	\$997	\$D197	\$858	\$D058	\$1114	\$D912
IF	\$AD	\$99A	\$D19A	\$85A	\$D05A	\$11CB	\$D9C9
RESTORE	\$AE	\$99C	\$D19C	\$85C	\$D05C	\$104B	\$D849
&	\$AF	\$9A3	\$D1A3	\$85E	\$D05E	\$03F5	\$03F5
GOSUB	\$B0	\$9A4	\$D1A4	\$860	\$D060	\$1123	\$D921
RETURN	\$B1	\$9A9	\$D1A9	\$862	\$D062	\$116D	\$D96B

COMMAND WORD	TOKEN	ADDRESS OF ASCII STRING		ADDRESS OF POINTER TO SUBROUTINE		ADDRESS OF SUBROUTINE	
		RAM	ROM	RAM	ROM	RAM	ROM
REM	\$B2	\$9AF	\$D1AF	\$864	\$D064	\$11DE	\$D9DC
STOP	\$B3	\$9B2	\$D1B2	\$866	\$D066	\$1070	\$D86E
ON	\$B4	\$9B6	\$D1B6	\$868	\$D068	\$11EE	\$D9EC
WAIT	\$B5	\$9B8	\$D1B8	\$86A	\$D06A	\$1F7B	\$E784
LOAD	\$B6	\$9BC	\$D1BC	\$86C	\$D06C	\$10CB	\$D8C9
SAVE	\$B7	\$9C0	\$D1C0	\$86E	\$D06E	\$10B2	\$D8B0
DEF	\$B8	\$9C4	\$D1C4	\$870	\$D070	\$10BC	\$E313
POKE	\$B9	\$9C7	\$D1C7	\$872	\$D072	\$1F72	\$E77B
PRINT	\$BA	\$9CB	\$D1CB	\$874	\$D074	\$12D7	\$DAD5
CONT	\$BB	\$9D0	\$D1D0	\$876	\$D076	\$1098	\$D896
LIST	\$BC	\$9D4	\$D1D4	\$878	\$D078	\$0EA7	\$D6A5
CLEAR	\$BD	\$9D8	\$D1D8	\$87A	\$D07A	\$0E6A	\$D66A
GET	\$BE	\$9DD	\$D1DD	\$87C	\$D07C	\$13A3	\$DBA0
NEW	\$BF	\$9E0	\$D1E0	\$87E	\$D07E	\$0E49	\$D649
TAB (	\$C0	\$9E3	\$D1E3				
TO	\$C1	\$9E7	\$D1E7				
FN	\$C2	\$9E9	\$D1E9				
SPC (	\$C3	\$9EB	\$D1EB				
THEN	\$C4	\$9EF	\$D1EF				
AT	\$C5	\$9F3	\$D1F3				
NOT	\$C6	\$9F5	\$D1F5				
STEP	\$C7	\$9F8	\$D1F8				
+	\$C8	\$9FC	\$D1FC	\$8B3	\$D0B3	\$1FB8	\$E7C1
-	\$C9	\$9FD	\$D1FD	\$8B6	\$D0B6	\$1FA1	\$E7AA
*	\$CA	\$9FE	\$D1FE	\$8B9	\$D0B9	\$2179	\$E982
/	\$CB	\$9FF	\$D1FF	\$8BC	\$D0BC	\$2260	\$EA69
^	\$CC	\$A00	\$D200	\$8BF	\$D0BF	\$268E	\$EE97
AND	\$CD	\$A01	\$D201	\$8C2	\$D0C2	\$1758	\$DF55
OR	\$CE	\$A04	\$D204	\$8C5	\$D0C5	\$1752	\$DF4F
>	\$CF	\$A06	\$D206	\$8C8	\$D0C8	\$26C7	\$EED0
=	\$D0	\$A07	\$D207	\$8CB	\$D0CB	\$169B	\$DE98
<	\$D1	\$A08	\$D208	\$8CE	\$D0CE	\$1768	\$DF65
SGN	\$D2	\$A09	\$D209	\$880	\$D080	\$2387	\$EB90
INT	\$D3	\$A0C	\$D20C	\$882	\$D082	\$241A	\$EC23
ABS	\$D4	\$A0F	\$D20F	\$884	\$D084	\$23A6	\$EBAF
USR	\$D5	\$A12	\$D212	\$886	\$D086	\$000A	\$000A
FRE	\$D6	\$A15	\$D215	\$888	\$D088	\$1AD7	\$E2DE
SCRN (	\$D7	\$A18	\$D218	\$88A	\$D08A	\$0C12	\$D412
PDL	\$D8	\$A1D	\$D21D	\$88C	\$D08C	\$17D0	\$DFCD
POS	\$D9	\$A20	\$D220	\$88E	\$D08E	\$1AF8	\$E2FF
SQR	\$DA	\$A23	\$D223	\$890	\$D090	\$2684	\$EE8D
RND	\$DB	\$A26	\$D226	\$892	\$D092	\$27A5	\$EFAE
LOG	\$DC	\$A29	\$D229	\$894	\$D094	\$2138	\$E941
EXP	\$DD	\$A2C	\$D22C	\$896	\$D096	\$2700	\$EF09
COS	\$DE	\$A2F	\$D22F	\$898	\$D098	\$27E1	\$EFEA
SIN	\$DF	\$A32	\$D232	\$89A	\$D09A	\$27E8	\$EFF1
TAN	\$E0	\$A35	\$D235	\$89C	\$D09C	\$2831	\$F03A
ATAN	\$E1	\$A38	\$D238	\$89E	\$D09E	\$2895	\$F09E
PEEK	\$E2	\$A3B	\$D23B	\$8A0	\$D0A0	\$1F5B	\$E764
LEN	\$E3	\$A3F	\$D23F	\$8A2	\$D0A2	\$1ECD	\$E6D6
STR\$	\$E4	\$A42	\$D242	\$8A4	\$D0A4	\$1BBE	\$E3C5
VAL	\$E5	\$A46	\$D246	\$8A6	\$D0A6	\$1EFE	\$E707
ASC	\$E6	\$A49	\$D249	\$8A8	\$D0A8	\$1EDC	\$E6E5
CHR\$	\$E7	\$A4C	\$D24C	\$8AA	\$D0AA	\$1E3D	\$E646
LEFT\$	\$E8	\$A50	\$D250	\$8AC	\$D0AC	\$1E51	\$E65A
RIGHT\$	\$E9	\$A55	\$D255	\$8AE	\$D0AE	\$1E7D	\$E686
MID\$	\$EA	\$A5B	\$D25B	\$8B0	\$D0B0	\$1E88	\$E691

Command Subroutines in the Cassette Version of APPLESOFT II

COMMAND WORD	ADDRESS OF ASCII STRING	ADDRESS OF POINTER TO SUBROUTINE	ADDRESS OF SUBROUTINE	COMMAND WORD	ADDRESS OF ASCII STRING	ADDRESS OF POINTER TO SUBROUTINE	ADDRESS OF SUBROUTINE
END	8D0	800	1072	LOAD	9BB	86C	10CB
FOR	8D3	802	F68	SAVE	9BF	86E	10B2
NEXT	8D6	804	14FC	DEF	9C3	870	1B0C
DATA	8DA	806	1197	POKE	9C6	872	1F72
INPUT	8DE	808	13B5	PRINT	9CA	874	12D7
DEL	8E3	80A	2B38	CONT	9CF	876	1098
DIM	8E6	80C	17DC	LIST	9D3	878	EA7
READ	8E9	80E	13E5	CLEAR	9D7	87A	E6A
GR	8ED	810	2B8C	GET	9DC	87C	13A3
TEXT	8EF	812	2B95	NEW	9DF	87E	E49
PR#	8F3	814	29EC	TAB(	9E2		
IN#	8F6	816	29E5	TO	9E6		
CALL	8F9	818	29DC	FN	9E8		
PLOT	8FD	81A	2A2C	SPC(	9EA		
HLIN	901	81C	2A39	THEN	9EE		
VLIN	905	81E	2A48	AT	9F2		
HGR2	909	820	2BD4	NOT	9F4		
HGR	90D	822	2BDE	STEP	9F7		
HCOLOR	910	824	2EE5	+	9FB	8B2	1FB8
HPLOT	916	826	2EFA	-	9FC	8B5	1FA1
DRAW	91B	828	2F62	*	9FD	8B8	2179
XDRAW	91F	82A	2F68	/	9FE	8BB	2260
HTAB	924	82C	2FE0	^	9FF	8BE	268E
HOME	928	82E	FC58	AND	A00	8C1	1758
ROT	92C	830	2F1A	OR	A03	8C4	1752
SCALE	930	832	2F20	>	A05	8C7	26C7
SHLOAD	936	834	2F6E	=	A06	8CA	169B
TRACE	93C	836	2A74	<	A07	8CD	1768
NOTRACE	941	838	2A76	SGN	A08	880	2387
NORMAL	948	83A	2A7A	INT	A0B	882	241A
INVERSE	94E	83C	2A7E	ABS	A0E	884	23A6
FLASH	955	83E	2A87	USR	A11	886	000A
COLOR=	95A	840	2A56	FRE	A14	888	1AD7
POP	960	842	116D	SCRN	A17	88A	C12
VTAB	963	844	2A5D	PDL	A1C	88C	17D0
HIMEM:	967	846	2A8D	POS	A1F	88E	1AF8
LOMEM:	96D	848	2AAD	SQR	A22	890	2684
ONERR	973	84A	2AD2	RND	A25	892	27A5
RESUME	978	84C	2B1F	LOG	A28	894	2138
RECALL	97E	84E	2BB8	EXP	A2B	896	2700
STORE	984	850	2B9B	COS	A2E	898	27E1
SPEED=	989	852	2A69	SIN	A31	89A	27E8
LET	98F	854	1248	TAN	A34	89C	2831
GOTO	992	856	1140	ATAN	A37	89E	2895
RUN	996	858	1114	PEEK	A3A	8A0	1F5B
IF	999	85A	11CB	LEN	A3E	8A2	1ECD
RESTORE	99B	85C	104B	STR\$	A41	8A4	1BBE
G	9A2	85E	800-see note	VAL	A45	8A6	1EFE
GOSUB	9A3	860	1123	ASC	A48	8A8	1EDC
RETURN	9A8	862	116D	CHR\$	A4B	8AA	1E3D
REM	9AE	864	11DE	LEFT\$	A4F	8AC	1E51
STOP	9B1	866	1070	RIGHT\$	A54	8AE	1E7D
ON	9B5	868	11EE	MID\$	A5A	8B0	1E88
WAIT	9B7	86A	1F7B				

NOTE: The \$ command is apparently there for later expansion and if you use it now your program will bomb!

MICROSOFT BASIC COMPILER

The Microsoft BASIC Compiler package contains the following software: BASIC Compiler, MACRO-80 assembler, and LINK-80 loader. The following manuals are also supplied: BASIC-80 Reference Manual, BASIC Compiler User's Manual, Utility Software Manual. The Utility Software Manual is the reference manual for MACRO-80 and LINK-80. The BASIC Compiler User's Manual describes the use of the compiler, its command format, compilation switches and error messages. The BASIC language that is used with the Microsoft BASIC Compiler is the same as described in this manual for Disk BASIC-80 with the following exceptions:

OPERATIONAL DIFFERENCES REQUIRED FOR COMPILING (LIMITATIONS)

The Compiler interacts with the console only to read compiler commands. These specify what files are to be compiled. There is no "direct mode," as with the BASIC-80 interpreter. Commands that are usually issued in the direct mode with the BASIC-80 interpreter are not implemented on the Compiler. The following commands are not implemented and will generate a warning message: AUTO, CLEAR, CLOAD, CSAVE, CONT, DELETE, EDIT, LIST, LLIST, RENUM, RUN, SAVE. LOAD, MERGE, and NEW also are not implemented and will cause fatal errors.

Because there is no direct mode for typing in programs or edit mode for editing programs, use Microsoft's EDIT-80 Text Editor or BASIC-80 interpreter for creating and editing programs. If you use the interpreter, be sure to SAVE the file with the A option.

The compiler cannot accept a physical line that is more than 127 characters in length. A logical statement, however, may contain as many physical lines as desired. Use line feed to start a new physical line within a logical statement.

To reduce the size of the compiled program, there are no program line numbers included in the object code generated by the compiler unless the /D, /X, or /E switch is set in the compiler command. Error messages, therefore, contain the address where the error occurred, instead of a line number. The compiler listing and the map generated by LINK-80 are used to identify the line that has the error. It is always a good idea to debug programs using the BASIC-80 interpreter before attempting to compile them. See the BASIC Compiler User's Manual for more information.

LANGUAGE DIFFERENCES

Most programs that run on the Microsoft BASIC-80 interpreter will run on the BASIC Compiler with little or no change. However, it is necessary to note differences in the use of the following program statements:

1. CALL

The <variable name> field in the CALL statement must contain an External symbol, i.e., one that is recognized by LINK-80 as a global symbol. This routine must be supplied by the user as an assembly language subroutine or a routine from the FORTRAN-80 library.

2. CHAIN and COMMON

The CHAIN and COMMON statements are not implemented on the compiler. They will generate a fatal error. COMMON will be implemented in a future release of the BASIC Compiler.

3. DEFINT/SNG/DBL/STR

The compiler does not "execute" DEFxxx statements; it reacts to the static occurrence of these statements, regardless of the order in which program lines are executed. A DEFxxx statement takes effect as soon as its line is encountered. Once the type has been defined for a given variable, it remains in effect until the end of the program or until a different DEFxxx statement with that variable takes effect.

4. DIM and ERASE

The DIM statement is similar to the DEFxxx statement in that it is scanned rather than executed. That is, DIM takes effect when its line is encountered. If the default dimension (10) has already been established for an array variable and that variable is later encountered in a DIM statement, a "Redimensioned array" error results. There is no ERASE statement in the compiler, so arrays cannot be erased and redimensioned. An ERASE statement will produce a fatal error.

Also note that the values of the subscripts in a DIM statement must be integer constants; they may not be variables, arithmetic expressions, or floating point values. For example,

DIM A1(I)

DIM A1(3 + 4)

are both illegal

5. END

During execution of a compiled program, an END statement closes files and returns control to the operating system. The compiler assumes an END statement at the end of the program, so "running of the end" produces proper program termination.

6. ON ERROR GOTO/RESUME <line number>

If a program contains ON ERROR GOTO and RESUME <line number> statements, the /E compilation switch must be used. If the RESUME NEXT, RESUME, or RESUME 0 form is used, the /X switch must also be included. See the BASIC Compiler User's Manual for an explanation of these switches.

7. REM

REM statements or remarks starting with a single quotation mark do not take up time or space during execution, and so may be used as freely as desired.

8. STOP

The STOP statement is identical to the END statement. Open files are closed and control returns to the operating system.

9. TRON/TROFF

In order to use TRON/TROFF, the /D compilation switch must be used. Otherwise, TRON and TROFF are ignored and a warning message is generated.

10. FOR/NEXT and WHILE/WEND

FOR/NEXT and WHILE/WEND loops must be statically nested.

11. Double Precision Transcendental Functions

SIN, COS, TAN, ATN, SQR, LOG, and EXP return double precision results if given a double precision argument. Exponentiation with double precision operands will return a double precision result.

EXPRESSION EVALUATION

During expression evaluation, the operands of each operator are converted to the same type, that of the most precise operand. For example,

$$QR = J\% + A! + Q\#$$

causes J% to be converted to single precision and added to A!. This result is converted to double precision and added to Q#

The Compiler is more limited than the interpreter in the handling numeric overflow. For example, when run on the interpreter the following program

```
I% = 20000
J% = 20000
K% = - 30000
M% = I% + J% - K%
```

yields 10000 for M%. That is, it adds I% to J% and, because the number is too large, it converts the result into a floating point number. K% is then converted to floating point and subtracted. The result of 10000 is found, and is converted back to integer and saved as M%.

The compiler, however, must make type conversion decisions during compilation. It cannot defer until the actual values are known. Thus, the compiler would generate code to perform the entire operation in integer mode. If the /D switch were set, the error would be detected. Otherwise, an incorrect answer would be produced.

In order to produce optimum efficiency in the compiled program, the compiler may perform any number of valid algebraic transformations before generating the code. For example, the program

```
I% = 20000
J% = - 18000
K% = 20000
M% = I% + J% + K%
```

could produce an incorrect result when run. If the compiler actually per-

forms the arithmetic in the order shown, no overflow occurs. However, if the compiler performs $I\% + K\%$ first and then adds $J\%$, an overflow will occur. The compiler follows the rules for operator precedence and parenthetical modification of such precedence, but no other guarantee of evaluation order can be made.

INTEGER VARIABLES

In order to produce the fastest and most compact object code possible, make maximum use of integer variables. For example, this program

```
FOR I = 1 TO 10  
  A(I) = 0  
NEXT I
```

can execute approximately 30 times faster by simply substituting "I%" for "I". It is especially advantageous to use integer variables to compute array subscripts. The generated code is significantly faster and more compact.

PASCAL

PASCAL

SYSTEM INFORMATION

VOLUME NAMES

A volume name consists of up to 7 characters followed by colon (:)/or pound sign (#) and a device number followed by a colon (:).

Valid Volume names —

APPLE3:, #4:, PRINTER: (for the print device only)

FILE NAMES

A file name can consist of up to 15 characters. File names for program source and code files consist of up to 10 characters and a 5 character suffix of either '.TEXT' (for program textfiles) or '.CODE' (for codefiles). Any character is valid for a file name but you should avoid using the \$, [, = , ?, RETURN and the control characters C, F, M, S, U and @ as these cannot be used from the keyboard.

RECOMMENDED SUFFIX CONVENTIONS FOR FILENAMES

Suffix	File Type
.TEXT	Program source files and human readable text
.CODE	Machine executable code
.DATA	Data files
.BAD	Bad blocks on disk. Normally only created by the E(xamine option in the Filer to mark a physically damaged area on the disk with an immovable file to allow continued use of the disk without risking storage of files in the damaged area.
.INFO	(reserved, not used at present)
.GRAF	(reserved, not used at present)
.FOTO	(reserved, not used at present)

FILE SPECIFICATION

A complete file specification consists of two parts, a volume name (or number) and a file name. The P(refix option of the Filer can be used to specify a default volume name to use with a file name.

Default volume specifications —

'.' — specifies use of the default volume Prefix.

'*' — specifies use of the boot disk as the default volume.

Filename shorthand and wildcard specifications —

'\$' — means 'use the same name here as in the previous file specification. This is used in the C(ompile, A(ssemble and T(ransfer options to specify the output file has the same name as the input file.

'=' — a wildcard specification. '=' matches any character or string of characters in the file name. Used in the Filer to change a group of files at one time. Typical uses might be to delete a group of files such as anything ending with the suffix '.TEXT'.

'?' — a wild card specification. '?' works the same as '=' except the Filer requests verification before affecting the file in any way.

Wildcard formats —

string1 = string2 or string1 ? string2

Where string1 and string2 are character strings. Either or both strings may be length zero.

Examples —

APPLE1: = TEXT Finds all files on APPLE1 that end with 'TEXT'.

NEW = .CODE Finds all files on the default volume that begin with 'NEW' and end with '.CODE'.

VOL1: = Finds all files on VOL1. This actually is the same as saying 'VOL1:' but has the effect of acting on a file by file basis rather than considering the disk as a whole block.

*:?.CODE Finds all files on the Boot volume with the suffix '.CODE' and asks for verification each time before doing anything to the file.

THE WORKFILE

The System workfile is kept on the boot disk. It consists of two parts, a text part and a code part. These parts are stored as two separate files, SYSTEM.WRK.TEXT and SYSTEM.WRK.CODE. The G(et) option in the Filer is used to specify the source file to use as the current workfile.

COMMANDS AVAILABLE AT ALL LEVELS

Certain commands can be executed at any level of the operating system regardless of what option is in effect at the moment. Note — these commands will only be detected when the next input is requested but will not affect the input line in any way (unless that is their purpose).

CTRL-A Shows the other 40-character "page" of the Apple Pascal system's 80-character display. This allows you to switch back and forth between the left and right hand sides of the display. (Note - this will not have any effect if you are using an external terminal or an 80-column display board in your Apple. Some 80-column display boards use CTRL-A to provide Shift and Shift Lock for upper and lower case.)

CTRL-Z	Initiates "Auto-follow" mode. The screen scrolls right and left to follow the cursor. Cancelled by CTRL-A and many other commands. (Note - will not work with external terminals and 80-column display boards.)
CTRL-@	Causes current program to be interrupted and issues the message, PROGRAM INTERRUPTED BY USER. Press spacebar to reinitialize the system.
CTRL-F	Causes subsequent program output to be flushed. The program continues but no output will be sent to the screen or the printer. It is cancelled by sending another CTRL-F.
CTRL-S	Stops any ongoing operating system process or program. When the next CTRL-S is typed, the process continues.
UPPER AND LOWER CASE (Pascal version 1.1 only)	
CTRL-E	Enable lower case mode. Upper case letters are displayed as inverse video. Changes the case of the keyboard from upper to lower case and vice versa. Thus it acts like a 'shift lock' key on a typewriter.
CTRL-W	Forces the next character to be upper case. Mainly used to capitalize the beginning of words. After the next character is typed the keyboard is forced into lower case.
CTRL-R	Turns the reverse video mode on. It does not change the case, all letters typed are still upper case unless CTRL-E or CTRL-W is used to change the the case.
CTRL-T	Turns off the reverse video mode and forces the keyboard into upper case mode.

COMMAND LEVEL

1. The Command level is reached automatically, each time the system is booted, RESET, or initialized. It is also reached when any program, including part of the operating system is terminated.
2. Use the Command level options to select any of the main subdivisions of the Apple Pascal operating system.

Command Level Options:

A(ssemble	Invokes the Assembler, which converts the text of an assembly language subroutine (found in the workfile or other specified textfile) into 6502 machine language.
-----------	---

C(ompile	Invokes the Pascal compiler, which converts the text of a Pascal program (found in the workfile or other specified textfile) into executable P-code.
D(ebug	Not implemented, do not use this option.
E(dit	Invokes the Editor, which is used to create and modify text. Reads the workfile or other specified file into the Apple for editing.
F(iler	Invokes the Filer, which is used to save, move and retrieve information stored on the diskettes.
H(alt	Does a 'cold boot' of the system, like turning the Apple's power off then on again.
I(nitalize	Does a 'warm boot' of the system, reloads the Pascal operating system but does not reload the P-code interpreter.
L(ink	Combines external P-code and machine language subroutines (found in the SYSTEM.LIBRARY or other specified library codefiles) into a Pascal host program (found in the workfile or other specified codefile).
M(ake	Used to start creation of an exec file. Any commands typed thereafter will be added to the exec file until the exec file termination character is typed.
R(un	Executes the current workfile, automatically compiling and linking (from SYSTEM.LIBRARY) first as necessary.
S(wapping	Toggles swapping mode off and on. Used to make more program space available to the user. Note - if swapping option is turned on, the boot diskette may not be removed from drive 1. The only exception is when the Filer is used. Then the entire operating system is in memory and the boot diskette may be exchanged temporarily for other diskettes.
U(ser restart	Attempts to execute the last program or option that was executed.
X(ecute	Loads and runs the specified Pascal program codefile.

FILER

1. From Command level, select F(iler. When Filer prompt line appears you may remove your boot diskette if necessary.
2. Use Filer commands to move, save and erase the workfile and other disk files.

FILER COMMANDS

- B(ad-blocks):** Tests all 280 blocks on the specified diskette to see that information has been recorded consistently. Bad blocks are reported. Use **X(amine** to fix bad blocks.
- C(hange:** Renames a diskette or a diskette file.
- D(ate:** Sets new current date for the system.
- E(xtended-directory-list:** Shows contents of a diskette, displaying extra information about the files and unused portions.
- G(et:** Designates a specified diskette file as the workfile. The next **E(dit**, **C(ompile** or **R(un** will use this file.
- K(runch:** Packs the files on a diskette so that unused portions of the diskette are combined into one area.
- L(ist-directory:** Shows the contents of a diskette.
- M(ake:** Creates a diskette directory entry with the specified filename. Produces a "dummy" file on the diskette.
- N(ew:** Clears the workfile, removing all **SYSTEM.WRK** files from the boot diskette.
- P(refix:** Changes the current default volume name to the volume name specified. If volume is specified as a unit number (ie. '#15') the prefix will be whatever volume is in that unit, regardless of the volume name.
- Q(uit:** Leaves the Filer and returns to the outermost Command level.
- R(emove:** Removes the specified file from a diskette's directory.
- S(ave:** Saves the workfile **SYSTEM.WRK** under the specified filename.
- T(ransfer:** Transfers information from one file to another file. Can be used to move or save diskette files, copy entire diskettes, or send files to printer or other device.
- V(olume:** Shows the devices and diskettes currently in the system, by volume number and by volume name.
- W(hat:** Tells the name and state (saved or not) of the workfile.
- X(amine:** Attempts to fix diskette blocks reported bad by the **B(ad-blocks** command. Marks blocks that can't be fixed.
- Z(ero:** Erases the directory of a diskette.

EDITOR

1. From Command level, select **F(iler**. Start a **N(ew** file or **G(et** an old file for re-editing (one-drive note: first **T(ransfer** the old file onto your boot diskette). **Q(uit** the Filer.
2. From Command level, select **E(ditor**. Press RETURN key if you are beginning a new file. Use the Editor commands to **I(nsert**, **D(elete**, **X(change** and move text. When you are through **Q(uit** and **U(pdate** the workfile.
3. If this is a program, you may **R(un** it now. Repeat steps 2 and 3 until the program runs correctly.

4. From the Command level, select the Filer. You may now Save the workfile.

EDITOR COMMANDS

Screen Commands:

CTRL-A	Shows the other 40-character "page" of the display.
CTRL-Z	Screen scrolls right and left to follow the cursor.

Special Characters:

CTRL-K	Produces left bracket: [
SHIFT-M	Produces right bracket:]

Cursor Moves:

CTRL-L	Moves repeat-factor lines down.
CTRL-O	Moves repeat-factor lines up.
right-arrow key	Moves repeat-factor spaces right.
left-arrow key	Moves repeat-factor spaces left.
spacebar	Moves repeat-factor spaces in set direction.
CTRL-I	Moves repeat-factor tab positions in set direction.
RETURN key	Moves to start of line that is repeat-factor lines away, in set direction.
=	Moves to start of latest text found, replaced or inserted.

Repeat-Factor:

Any integer typed before a move or command. If repeat-factor is / the move or command is repeated as many times as possible in the file.

Set Direction:

, <	All change set direction to backward
. >	All change set direction to forward

Moving Commands:

J(ump:	Jumps to file's B(eginning or E(nd, or to previously set Marker.
P(age:	Moves the cursor repeat-factor pages in the set direction.
F(ind:	Looks in the set direction for the repeat-factor-th L(iteral or T(oken occurrence of the <TARGET> string, which must be typed with delimiters. S means use the same string as before.

Text Changing Commands

I(nsert:	Inserts text. Use left-arrow key to backspace over insertion. CTRL-Q deletes back to the most recent RETURN character in the current insertion. CTRL-X acts like CTRL-Q except that it also deletes the RETURN.
D(elete:	Deletes all text moved over by the cursor. Back up cursor to undelete.
Z(ap:	Deletes all text between the current cursor position and the "equals position" (at the start of the latest text found, replaced or inserted).
C(opy:	Copies a diskette file, or what was last inserted, deleted or zapped, into the file at the position of the cursor.
X(change:	Replaces the character under the cursor with the character typed. Each line must be done separately. Pressing the left-arrow key causes the original character to re-appear.
R(eplace:	Looks in the set direction for the next L(iteral or T(oken occurrence of <TARG> string, and replaces it with <SUB> string. Continues repeat-factor times. Both strings must be typed with delimiters. V(erify option asks for permission to replace. S means use the same <TARG> or the same <SUB> string as before.

Formatting Commands

A(djust:	Adjusts indentation of the line the cursor is on. Left-arrow and right-arrow keys move the line left and right. Moving cursor up or down adjusts lines above or below by same amount.
M(argin:	Starting at the cursor position, adjusts all text between two blank lines (one paragraph) to the margins which have been set.

Miscellaneous Commands

S(et:	Sets a M(arker of the specified name at the current cursor position. Sets options in the E(nvironment for A(uto-indent, F(illing, margins, default search mode, and C(ommand characters.
V(erify:	Redisplays the screen with the cursor centered.
Q(uit:	Leaves the Editor. You may U(pdate the workfile, E(xit without updating, R(eturn to the Editor, W(rite to any diskette file, or S(ave to your original file.

COMPILER

To use:

1. From Command level, select R(un or Compile.
2. If a text workfile exists, that file is compiled automatically. Otherwise you are prompted to specify a source textfile and then specify a destination codefile (use '\$' wildcard to specify same name as textfile).
3. If the compiler finds an error, select the E(ditor to fix it.

COMPILER OPTIONS

All Compiler options are placed in "dollar-sign" comments:

(* \$option*)

Examples: (* \$G-*) (* \$!TURTLE.TEXT*)

Compiler-option specification may be combined in one comment:

(* \$option,option*)

Example: (* \$SF-,S+,G+*)

C	Following characters are placed directly into codefile.
G+	Allows GOTO statements
*G-	Forbids GOTO statements
*I+	Generates I/O-checking code.
I-	No I/O checking.
Ifilename	Includes named source file in compilation.
P	Pages listing.
Q+	Suppress screen message.
*Q-	Sends procedure names and line numbers to CONSOLE:
R+	Generates range-checking code for array subscripts and variables.
R-	No range checking.
S+	Puts Compiler in swapping mode.
S++	Compiler does even more swapping.
*S-	Non-swapping mode: entire Compiler in memory.
*V+	Turns String length checking for Var parameters on
V-	Turns String length checking for Var parameters off
*U+	Compiles on user lex level.
U-	Compiles on system lex level.
Ufilename	Specifies name of library file for finding UNITS.
* — indicates default options for compiler	

SYSTEM FUNCTIONS AND PROCEDURES

String Built-Ins

Integer-Valued Functions:

LENGTH (STRG) returns length of string

POS (SUBSTRG , STRG) returns index of first occurrence of SUBSTRG within STRG.

String-Valued Functions:

CONCAT (STRGs) returns concatenation of string.

COPY (STRG , INDES , COUNT) returns a substring of STRG.

Procedures:

DELETE (STRG , INDES , COUNT) deletes a substring of STRG.

INSERT (SUBSTRG , STRG , INDEX) inserts a substring into STRG.

STR (LONG , STRG) converts integer or long integer to string of decimal digits.

Input and Output Built-Ins

Opening and Closing Files:

RESET (FILEID [, TITLE]) opens existing diskette file, or resets pointers to beginning if already open.

REWRITE (FILEID [, TITLE]) opens new diskette file.

CLOSE (FILEID [, OPTION]) closes file. OPTION may be LOCK, NORMAL, PURGE, or CRUNCH. Default is NORMAL.

File Pointer Status:

EOF ([FILEID]) boolean, true if end of file has been reached or file is closed. Default FILEID is INPUT.

EOLN ([FILEID]) boolean, true if end of line has been reached. Default FILEID is INPUT.

SEEK (FILEID , INTEGER) moves file pointer to specified record.

Typed File I/O:

GET (FILEID) reads current file record into window & advances file pointer. Default FILEID is INPUT.

PUT (FILEID) writes window into current file record & advances file pointer. Default FILEID is OUTPUT.

IORESULT returns an integer value which depends on status of most recent I/O operation. Value is zero for OK completion.

READ ([FILEID,] VBLs) where VBLs means one or more variables separated by commas. Successive values are read from file into variables. Default FILEID is INPUT. FILEID must be ofk type TEXT (FILE OF CHAR) or INTERACTIVE.

READLN ([FILEID,] VBLs) Like READ, but skips to beginning of next line after reading value for last VBL.

WRITE ([FILEID,] [EXPRs]) where EXPRs means one or more expressions separated by commas. Each EXPR may also be followed by field width and number of decimal places. Expression values are written to successive file records. Default FILEID is OUTPUT. FILEID must be of type TEXT (FILE OF CHAR) or INTERACTIVE.

WRITELN [([FILEID,] [EXPRs])] Like WRITE, but writes an end-of-line after last EXPR value.

PAGE (FILEID) writes a top-of-form (ASCII 12).

Device I/O:

UNITREAD (UNITNUMBER , ARRAY , LENGTH [, [BLOCKNUMBER]
[, MODE]])
UNITWRITE (UNITNUMBER , ARRAY , LENGTH [, [BLOCKNUMBER]
[, MODE]])
UNITBUSY (UNITNUMBER) : BOOLEAN
UNITWAIT (UNITNUMBER)
UNITCLEAR (UNITNUMBER)

Untyped File I/O:

BLOCKREAD (FILEID, ARRAY, BLOCKS [, RELBLOCK]) : INTEGER
BLOCKWRITE (FILEID, ARRAY, BLOCKS [, RELBLOCK]) : INTEGER

Miscellaneous Built-Ins

ATAN (NUMBER) returns a REAL value. This is the ARCTAN function of Standard Pascal. NUMBER may be REAL or INTEGER.
LOG (NUMBER) returns a REAL value, the log base 10 of NUMBER. NUMBER may be REAL or INTEGER.
TRUNC (NUMBER) returns an INTEGER value. This is like Standard Pascal except that NUMBER may be LONG INTEGER instead of REAL. NUMBER may not exceed MAXINT.
PWROFTEN (EXPONENT) returns a REAL value which is 10 to the EXPONENT power. EXPONENT is an INTEGER in the range 0..37.
MARK (HEAPPTR) where HEAPPTR is of type INTEGER, HEAPPTR is called by name and is set to current top-of-heap.
RELEASE (HEAPPTR) where HEAPPTR is of type INTEGER. The current top-of-heap pointer is set to HEAPPTR.
HALT causes non-fatal run-time error; halts program.
EXIT causes orderly exit from procedure, function, or program.
MEMAVAIL returns an INTEGER value, the number of words between top-of-stack and top-of-heap.
GOTOXY (XCOORD , YCOORD) moves screen cursor to specified coordinates. XCOORD is an INTEGER in the range 0..79 and YCOORD is an INTEGER in the range 0..23.
TREESEARCH (ROOTPTR , NODEPTR , NAME) searches for NAME in a binary tree.

Byte-Oriented Built-ins

SIZEOF (VARIABLE OR TYPE IDENTIFIER)
SCAN (LIMIT , PEXPR , SOURCE)
MOVELEFT (SOURCE , DESTINATION , COUNT)
MOVERIGHT (SOURCE , DESTINATION , COUNT)
FILLCHAR (DESTINATION , COUNT , CHARACTER)

SYNTAX ERRORS IN APPLE UCSD PASCAL

The Apple will normally give the complete error message when you return to the Editor after finding a syntax error during compilation. The only time it won't have these messages is if you remove SYSTEM.SYNTAX from the boot diskette, syntax errors will still be reported to you, but by number only.

- 1: Error in simple type
- 2: Identifier expected
- 3: 'PROGRAM' expected
- 4: ')' expected
- 5: ':' expected
- 6: Illegal symbol
- 7: Error in parameter list
- 8: 'OF' expected
- 9: '(' expected
- 10: Error in type
- 11: '[' expected
- 12: ']' expected
- 13: 'END' expected
- 14: ';' expected
- 15: Integer expected
- 16: '=' expected
- 17: 'BEGIN' expected
- 18: Error in declaration part
- 19: error in field-list
- 20: '.' expected
- 21: '*' expected
- 22: 'Interface' expected
- 23: 'Implementation' expected
- 24: 'Unit' expected
- 50: Error in constant
- 51: ':=' expected
- 52: 'THEN' expected
- 53: 'UNTIL' expected
- 54: 'DO' expected
- 55: 'TO' or 'DOWNT0' expected in for statement
- 56: 'IF' expected
- 57: 'FILE' expected
- 58: Error in <factor> (bad expression)
- 59: Error in variable
- 101: Identifier declared twice
- 102: Low bound exceeds high bound
- 103: Identifier is not of the appropriate class
- 104: Undeclared identifier
- 105: sign not allowed
- 106: Number expected
- 107: Incompatible subrange types

- 108: File not allowed here
- 109: Type must not be real
- 110: <tagfield> type must be scalar or subrange
- 111: Incompatible with <tagfield> part
- 112: Index type must not be real
- 113: Index type must be a scalar or a subrange
- 114: Base type must not be real
- 115: Base type must be a scalar or subrange
- 116: Error in type of standard procedure parameter
- 117: Unsatisfied forward reference
- 118: Forward reference type identifier in variable declaration
- 119: Re-specified params not OK for a forward declared procedure
- 120: Function result type must be scalar, subrange or pointer
- 121: File value parameter not allowed
- 122: A forward declared function's result type can't be re-specified
- 123: Missing result type in function declaration
- 124: F-format for reals only
- 125: Error in type of standard procedure parameter
- 126: Number of parameters does not agree with declaration
- 127: Illegal parameter substitution
- 128: Result type does not agree with declaration
- 129: Type conflict of operands
- 130: Expression is not of set type
- 131: Tests on equality allowed only
- 132: Strict inclusion not allowed
- 133: File comparison not allowed
- 134: Illegal type of operand(s)
- 135: Type of operand must be boolean
- 136: Set element type must be scalar or subrange
- 137: Set element types must be compatible
- 138: Type of variable is not array
- 139: Index type is not compatible with the declaration
- 140: Type of variable is not record
- 141: Type of variable must be file or pointer
- 142: Illegal parameter solution
- 143: Illegal type of loop control variable
- 144: Illegal type of expression
- 145: Type conflict
- 146: Assignment of files not allowed
- 147: Label type incompatible with selecting expression
- 148: Subrange bounds must be scalar
- 149: Index type must be integer
- 150: Assignment to standard function is not allowed
- 151: Assignment to formal function is not allowed
- 152: No such field in this record
- 153: Type error in read
- 154: Actual parameter must be variable
- 155: Control variable cannot be formal or non-local

- 156: Multidefined case label
- 157: Too many cases in case statement
- 158: No such variant in this record
- 159: Real or string tagfields not allowed
- 160: Previous declaration was not forward
- 161: Again forward declared
- 162: Parameter size must be constant
- 163: 'Missing variant in declaration
- 164: Substition of standard proc/func not allowed
- 165: Multidefined label
- 166: Multideclared label
- 167: Undeclared label
- 168: Undefined label
- 169: Error in base set
- 170: Value parameter expected
- 171: Standard file was re-declared
- 172: Undeclared external file
- 174: Pascal function or procedure expected
- 175: Actual parameter max string length < var formal max length
- 182: Nested units not allowed
- 183: External declaration not allowed at this nesting level
- 184: External declaration not allowed in interface section
- 185: Segment declaration not allowed in unit
- 186: Labels not allowed in interface section
- 187: Attempt to open library unsuccessful
- 188: Unit not declared in previous uses declaration
- 189: 'Uses' not allowed at this nesting level
- 190: Unit not in library
- 191: No private files
- 192: 'Uses' must be in interface section
- 193: Not enough room for this operation
- 194: Comment must appear at top of program
- 195: Unit not importable
- 201: Error in real number - digit expected
- 202: String constant must not exceed source line
- 203: Integer constant exceeds range
- 204: 8 or 9 in octal number
- 250: Too many scopes of nested identifiers
- 251: Too many nested procedures or functions
- 252: Too many forward references of procedure entries
- 253: Procedure too long
- 254: Too many long constants in this procedure
- 256: Too many external references
- 257: Too any externals
- 258: Too many local files
- 259: Expression too complicated
- 300: Division by zero
- 301: No case provided for this value

- 302: Index expression out of bounds
 - 303: Value to be assigned is out of bounds
 - 304: Element expression out of range
 - 398: Implementation restriction
 - 399: Implementation restriction
 - 400: Illegal character in text
 - 401: Unexpected end of output
 - 402: Error in writing code file, not enough room
 - 403: Error in reading include file
 - 404: Error in writing list file, not enough room
 - 405: Call not allowed in separate procedure
 - 406: Include file not legal
 - 407: Too many libraries
 - 408: (*S + *) needed to compile units
-

ASSEMBLER

1. From Command level, select A(ssemble).
2. If a text workfile exists, that file is assembled automatically. Otherwise, you are prompted to specify a source textfile and then a destination codefile (use '\$' wildcard to specify same name as textfile).
3. Finally, you are prompted to specify an output textfile for the assembly listing if you want one.
4. If the Assembler finds an error, use the Editor to fix it.

ASSEMBLER DIRECTIVE SUMMARY

Metasymbol Notation

Square brackets [like this] surround optional elements which you may supply. Angle brackets <like this> surround required elements which you must supply. The metasymbol brackets and the brief definition at the end of each line are not to be typed.

Routine Delimiting Directives

.PROC	< identifier > [, expression]	Begins a procedure
.FUNC	< identifier > [, expression]	Begins a function
.END		Ends entire assembly

Label Definitions and Space-Allocation Directives

[label]	.ASCII	"< character string >"	Inserts ASCII of chars.
[label]	.BYTE	[valuelist]	Inserts byte of value.
[label]	.BLOCK	length [, value]	Inserts block of value.
[label]	.WORD	< valuelist >	Inserts word of value.
label	.EQU	< value >	Assigns value to label.
	.ORG	< value >	Next byte at start of assembly file + value.
	.ABSOLUTE		Precedes 1st .PROC; all .ORGs put next byte at abs. location = value.

.INTERP

1st loc. of interpreter, in relative-location expressions.

Macro Facility Directives

.MACRO <identifier>
.ENDM

Begins a macro definition
Ends a macro definition

Conditional Assembly Directives

[label] .IF <expression>

[.ELSE]

Begins condit'l assembly
If true, assembles next text [up to .ELSE];
if false, only text after a .ELSE .
Ends condit'l assembly.

.ENDC

Host-Communication Directives

.CONST <identifierlist>

.PUBLIC <identifierlist>

.PRIVATE <identifier[:integer] list>

Takes value from global const in Pascal host.
Uses a global variable from the Pascal host.
Variable not accessible to the Pascal host. Default :1 word/ident.

External Communication Directives

.DEF <identifierlist>

.REF <identifierlist>

Makes label available to other routines.
Label refers to another routine's .DEF'd label.

Listing Control Directives

.LIST and .NOLIST

.MACROLIST
and .NOMACROLIST

.PATCHLIST
and .NOPATCHLIST

.PAGE
.TITLE "<title>"

Turns assembly listing on and off.

Turns listing of macro expansions on and off.

Turns listing of backpatches on and off.
Puts page-feed in listing.
Titles each page of current .PROC or .FUNC .

File Directive

.INCLUDE <filename>

Includes named text file in the assembly.

ASSEMBLER SYNTAX ERRORS

When the 6502 Assembler discovers an error in your assembly-language routine, it gives an error message taken from the file 6500.ERRORS, usually found on diskette APPLE2: . If the file 6500.ERRORS is not available in any drive, errors will be reported by number, only.

The 6502 Assembler error message corresponding to each error number is given in the table below. Some people may prefer to gain some additional diskette space by removing the file 6500.ERRORS and using this table instead.

The first portion of this table lists all the general error messages. Machine errors specific to Apple's 6502 are found in the last portion of the table.

GENERAL ERRORS

- | | |
|-----|--|
| 1: | Undefined label |
| 2: | Operand out of range |
| 3: | Must have procedure name |
| 4: | Number of parameters expected |
| 5: | Extra garbage on line |
| 6: | Input line over 80 characters |
| 7: | Not enough .IF's |
| 8: | Must be declared in .ASECT before used |
| 9: | Identifier previously declared |
| 10: | Improper format |
| 11: | .EQU expected |
| 12: | Must .EQU before use if not to a label |
| 13: | Macro identifier expected |
| 14: | Word addressed machine |
| 15: | Backward .ORG currently not allowed |
| 16: | Identifier expected |
| 17: | Constant expected |
| 18: | Invalid structure |
| 19: | Extra special symbol |
| 20: | Branch too far |
| 21: | Variable not PC relative |
| 22: | Illegal macro parameter index |
| 23: | Not enough macro parameters |
| 24: | Operand not absolute |
| 25: | Illegal use of special symbols |
| 26: | Ill-formed expression |
| 27: | Not enough operands |
| 28: | Cannot handle this relative expression |
| 29: | Constant overflow |
| 30: | Illegal decimal constant |
| 31: | Illegal octal constant |
| 32: | Illegal binary constant |
| 33: | Invalid key word |
| 34: | Macro stack overflow - 5 nested limit |
| 35: | Include files may not be nested |

- 36: Unexpected end of input
- 37: This is a bad place for an `.INCLUDE` file
- 38: Only labels & comments may occupy col 1
- 39: Expected local label
- 40: Local label stack overflow
- 41: String constant must be on one line
- 42: String constant exceeds 80 characters
- 43: Illegal use of macro parameter
- 44: No local labels in `.ASECT`
- 45: Expected key word
- 46: String expected
- 47: Bad block, parity error (CRC)
- 48: Bad unit number
- 49: Bad mode, illegal operation
- 50: undefined hardware error
- 51: Lost unit, unit is no longer on-line
- 52: Lost file, file is no longer in director
- 53: Bad title, illegal file name
- 54: No room, insufficient space on disk
- 55: No unit, no such volume on-line
- 56: No file, no such file on volume
- 57: Duplicate file
- 58: Not closes, attempt to open an open file
- 59: Not open, attempt to access a closed fil
- 60: Bad format, error in reading real or int
- 61: Nested macro definitions illegal
- 62: '=' or ' ' expected
- 63: May not EQU to undefined labels
- 64: Must declare `.ABSOLUTE` before 1st `.PROC`

6502-SPECIFIC ERRORS

- 76: Index register required
- 77: 'X' or 'Y' expected
- 78: Zero-page address required
- 79: Illegal use of register
- 80: Index register expected
- 81: Ill-formed operand
- 82: 'X' expected for indexed addressing
- 83: Must use 'X' index register
- 500: General assembler error

LINKER

1. From Command level, select R(un or L(ink.
2. R(un links the compiled workfile into UNITs and routines found in `SYSTEM.LIBRARY`, automatically. L(ink prompts you to specify a host codefile and then specify as many library codefiles as needed. Press the RETURN key to stop giving library files.

3. Next, you are prompted to specify a map textfile for storing Linker. Normally, just press the RETURN key to go on.
4. Finally, you are prompted to specify and output codefile for the linked program. If you just press the RETURN key, the output code file will be stored in SYSTEM.WRK.CODE.

UTILITIES

1. From the Command level, select the X(ecute option. When you are prompted 'EXECUTE WHAT FILE?', type the name of a utility program.
2. The utility programs allow you to format new disks, put routines into a library file, change the system to use an external terminal, change the printer output and use your Apple as a calculator.

These are the utility programs provided on volume 'APPLE3:' —

FORMATTER	Prepares new diskettes for use by Apple Pascal
LIBRARY	Puts UNITS and routines into a library file.
LIBMAP	Shows the contents of a library file.
SETUP	Changes the system for use with an external terminal.
BINDER	Changes the system for use with an external terminal.
LINEFEED	Stops the sending of a linefeed after RETURN to printers.
CALC	Lets you add, subtract, multiply and divide numbers.

UTILITIES SUMMARY

Formatting New Diskettes

1. X(ecute APPLE3:FORMATTER
2. When asked FORMAT WHICH DISK? , put a new diskette in any drive and type that drive's volume number.
3. To quit, press the RETURN key in response to FORMAT WHICH DISK?

The System Librarian

1. X(ecute APPLE3:LIBRARY
2. When asked for an OUTPUT CODE FILE — > , type a filename for the new library file. E.g., MYDISK:NEW.LIBRARY
3. When asked for a LINK CODE FILE — > , type the name of the file which contains the first items to put in the new library. E.G., APPLE1:SYSTEM.LIBRARY
4. To transfer an item from the source Link Code File to the new library Output Code File, type the item's Link Code File slot number (0 to 15) and press the spacebar. When asked SLOT TO LINK INTO? , type the number of the slot you want the item to occupy in the Output Code File and press the spacebar.
5. Type N to begin taking items from a N(ew Link Code File.

6. When all desired items have been transferred to the new library, lock the new library by typing Q for Q(uit. When asked NOTICE? , type a copyright message or press RETURN.
7. To use the new library, it must be placed on your boot diskette and it must be named SYSTEM.LIBRARY

Library Mapping

1. X(ecute APPLE3:LIBMAP
2. When prompted to ENTER LIBRARY NAME: , type the name of the library or other code file whose contents you wish to see mapped. E.g., APPLE:SYSTEM.LIBRARY
3. When asked LIST LINKER INFO TABLE? , press the spacebar or RETURN key unless you want that information.
4. When prompted for a MAP OUTPUT FILE NAME: , type the name of the diskette file or other device to which you wish the map sent. Just pressing the RETURN key sends the map to CONSOLE: .
5. When prompted again to ENTER LIBRARY NAME: , type the name of the next library file whose contents you wish mapped, or press the RETURN key to quit the program.

System Reconfiguration

1. To use your system with an external terminal, make a copy of APPLE1: for use as your new boot diskette. Give this new boot diskette a different name, such as SOROC1: or BRIAN: .
2. If your terminal is a Soroc IQ120, T(ransfer APPLE3:SOROC.MISCINFO to your new boot diskette and change its filename to SYSTEM.MISCINFO

If your terminal is a Hazeltine 1500, T(ransfer APPLE3:HAZEL.MISCINFO to your new boot diskette and change its filename to SYSTEM.MISCINFO

If your terminal is neither of the above, X(ecute APPLE3:SETUP , and let the program T(each you how to C(hange the parameters to suit your terminal. When the parameters are set correctly, Q(uit and do a D(isk update. This creates the file NEW.MISCINFO on your boot diskette. Then E(xit the program. Finally, you must T(ransfer NEW.MISCINFO to your new boot diskette and change this file's name to SYSTEM.MISCINFO

3. Read the next section, which tells how to change the GOTOXY Pascal procedure to work correctly with your terminal.

Changing Gotoxy Communication

1. Read the previous section on reconfiguring your system to suit your external terminal.
2. X(ecute APPLE3:BINDER
3. When you are asked FILE WHICH CONTAINS GOTOXY? , type the name of the codefile containing the new GOTOXY procedure for your terminal. One-drive note: this file must be on your boot diskette.

If your terminal is a Soroc IQ120, type APPLE3:SOROCGOTO

If your terminal is a Hazeltine 1500, type APPLE3:HAZELGOTO

If your terminal is neither of the above, type the name of the codefile containing a new GOTOXY procedure that you modified from APPLE3:SOROCGOTO.TEXT to suit your terminal and then compiled.

4. This program creates the file NEW.PASCAL on your boot diskette. You must now T(ransfer NEW.PASCAL onto the new boot diskette created in the previous section, and change the name of this file to SYSTEM.PASCAL . Your system will discover the new files SYSTEM.MISCINFO and SYSTEM.PASCAL the next time you boot with your new boot diskette.

Removing Linefeed From Return

1. The Apple Pascal system automatically supplies a linefeed after every RETURN character sent to the printer. If your printer does not work correctly with this arrangement, X(ecute APPLE3:LINEFEED
2. Until the next boot, RESET, or initialization, no linefeed will be sent to the printer after RETURN characters.
3. You may wish to put LINEFEED.CODE on your boot diskette, and change its name to SYSTEM.STARTUP Then this utility will be executed automatically, each time the system is booted.

Calculator

1. X(ecute APPLE3:CALC
2. When you are prompted $\rightarrow$, type any mathematical expression involving decimal numbers with fewer than 36 digits, and the operators + - * / Use parentheses (and) to keep expressions unambiguous. Example:

$\rightarrow (2.32 + .029) * (-1.75 / 4394.17)$

3. Press the RETURN key to see the result of evaluating the expression, expressed in scientific notation with six digits and a power-of-ten indicator. Result of above example:

$-9.35501E-4$

In other notation, $-9.35501E-4 = -9.35501 * (10^{-4}) = -.000935501$

4. To quit, press RETURN when prompted $\rightarrow$

EXECUTION ERROR MESSAGES

When a program or any portion of the Apple Pascal operating system is running, execution errors are reported by number or by message, in one of the following forms:

EXEC ERR # 10

S# 1, P# 7, I# 56

TYPE <SPACE> TO CONTINUE

IO ERROR: VOL NOT FOUND

S# 1, P# 7, I# 56

TYPE <SPACE> TO CONTINUE

where S# specifies the program's current segment number, P# specifies the procedure number within that segment, and I# specifies the byte

number in that procedure where the error was detected. User I/O errors (only) are reported in the more detailed second form only if file SYSTEM.PASCAL is in the boot drive at the time.

See the Apple Pascal Operating System Reference Manual's section on error handling, in the appendix ARCHITECTURE OF THE P-MACHINE, for more details. Also see the Apple Pascal Language Reference Manual's discussion of the L+ (compiled listing) compiler option, which describes how to list segment, procedure, and byte number information when you compile a program.

Error Number	Error Message and Description	Fatal Error?
Ø	System error of undefined nature.	FATAL
1	Invalid index, value out of range for string or subrange (XINVNDX). Does not occur if R- compiler option used.	
2.	No segment: bad code file (XNOPROC). File reads correctly from disk, but not a valid segment.	
3.	Procedure not present at exit time (XNOEXIT): exit from a procedure that was not previously called or active.	
4.	Stack overflow (XSTKOVVR): the program stack and the heap together have exceeded available user memory.	
5.	Integer overflow (XINTOVR). Integer arithmetic gave a result >16 bits. Long integer arithmetic gave an intermediate result >36 digits or final result was assigned to variable of insufficient size.	
6.	Divide by zero (XDIVZER).	
7.	Invalid memory reference <bus timed out> (XBAD-MAM): (not used on the Apple).	
8.	User break (XUBREAK: "break" key pressed (CTRL-@, on the Apple).	
9.	System I/O error (XSYIOER): error in attempting to read an operating system segment from disk.	FATAL
10.	User I/O error (XUIOERR): error when user's program attempted a blockread, blockwrite, get, or put. If file SYSTEM.PASCAL available, this error is further reported as an I/O ERROR (see next page).	
11.	Unimplemented instruction (XNOTIMP): op-code not implemented, or CSP to non-existent routine.	
12.	Floating point math error (XFPIERR): error in real number format, overflow, underflow, etc.	
13.	String to long (XS2LONG): attempt to store a source string into a destination string of insufficient size.	
14.	Halt, Breakpoint (without debugger in core) (XHLTBPT): (not used on the Apple).	

15. Bad Block (not use on the Apple; Apple reports I/O ERROR #64, instead).

A fatal error either causes the system to "cold boot" itself or (if the error was totally lethal to the system) forces you to "cold boot" the system by turning the Apple off and then on again. All other errors cause the system to re-initialize itself (do a "warm boot" by calling system procedure INITIALIZE), usually after you press the Apple's spacebar to continue.

I/O ERROR MESSAGES

Error Number	Error Message and Description
Ø	No error
1	Diskette has bad block: parity error (CRC). (not used on the Apple.)
2.	Bad device (volume) number.
3.	Bad mode: illegal operation. (For example, an attempt to read from PRINTER;.)
4.	Undefined hardware error. (Not used on the Apple.)
5.	Lost device: device is no longer on-line, after successfully starting an operation using that device.
6.	Lost file: file is no longer in the diskette directory, after successfully starting an operation using that file.
7.	Bad title: illegal filename. (For example, filename is more than 15 characters long.)
8.	No room: insufficient space on the specified diskette. (Files must be stored in contiguous diskette blocks.)
9.	No device: the specified volume is not on-line.
10.	No file: the specified file is not in the directory of the specified volume.
11.	Duplicate file: attempt to re-write a file when a file of that name already exists.
12.	Not closed: attempt to open an already-open file.
13.	Not open: attempt to access a closed file.
14.	Bad format: error in reading real or integer. (For example, your program expects an integer input but you typed a character.)
15.	Ring buffer overflow: characters are arriving at the Apple faster than the input buffer can accept them.
16.	Write-protect error: the specified diskette is write-protected.
64.	Device error: failed to complete a read or write correctly (bad address or data field on diskette)

The appropriate one of the I/O error messages is given when execution error #10 occurs if the file SYSTEM.PASCAL is in the boot drive. See the Apple Pascal Language Reference Manual for information about the Apple Pascal function IORESULT, which returns the error numbers shown above.

DEVICE VOLUME NUMBERS AND NAMES

Volume Number	Volume Name	Description of Input/Output Device
#0:		(not used)
#1	CONSOLE:	Screen and keyboard with echo
#2	KEYBOARD:	Reads keyboard without echoing it
#3		(not used)
#4	diskette name:	Boot disk (slot 6, drive 1)
#5	diskette name:	2nd disk drive (slot 6, drive 2)
#6	PRINTER:	Printer
#7	REMIN:	Remote input
#8	REMOUT:	Remote output
#9	diskette name	Additional disk drive (slot 4, drive 1)
#10	diskette name	2nd disk drive (slot 6, drive 2)
#11	diskette name	Additional disk drive (slot 5, drive 1)
#12	diskette name	Additional disk drive (slot 5, drive 2)

Note: The V(olumes command reports volume #2: as SYSTEM:

APPLE I/O DEVICE SLOTS

In using an Apple computer with the Apple Pascal operating system, the Apple's peripheral equipment slots are assigned as follows:

Apple Input/Output Device and Card Slot	Assigned to That Slot	Apple Pascal Operating System Use
0	Apple Language Card *	Stores interpreter and I/O system PRINTER: or #6:
1	Printer (Communications Interface Card, Serial Interface Card, or Parallel Printer Interface Card)	
2	Modem, Apple-to-Apple communication, etc. (Communications Interface Card or Serial Interface Card)	REMIN: or #7: REMOUT: or #8:
3	External terminal (Communications Interface Card. Use of Serial Interface Card is tolerated.)	CONSOLE: or #1: (with echo on input) SYSTEM: or #2: (without echo on input)
4	5th disk drive (Drive 1) 6th disk drive (Drive 2) (Disk Controller Card)	diskette name: or #9: diskette name: or #10:
5	3rd disk drive (Drive 1) 4th disk drive (Drive 2) (Disk Controller Card)	diskette name: or #11: diskette name: or #12:
6	Boot disk drive (Drive 1) * 2nd disk drive (Drive 2) (Disk Controller Card)	diskette name: or #4: diskette name: or #5:

- 7 Must not contain a disk drive
(Do not install a Disk
Controller Card here.)
Apple keyboard and screen
(only if there is no Communications
Interface Card in slot #3)

CONSOLE: or #1:
(with echo on input)
SYSTEM: or #2:
(without echo on input)

Note: An asterisk (*) indicates a device which is REQUIRED to be present in that slot.

Note: It is possible to use the Apple Pascal operating system with an Apple computer system containing only a single disk drive.

Note: If a Communications Interface Card is installed in slot #3, for use with an external terminal, booting the system automatically loads the Apple Pascal operating system into a different area of memory (using some of the Apple's text-screen memory). This happens whether or not the external terminal is actually connected to the Communications Interface Card. To return to using the Apple without the external terminal, you must turn off the Apple's power switch, remove the Communications Interface Card from slot #3, and turn on the Apple's power again to re-boot the system.

RESERVED WORDS IN PASCAL

STANDARD PASCAL RESERVED WORDS

AND	MOD
ARRAY	NIL
BEGIN	NOT
BOOLEAN	OF
CASE	OR
CHAR	PACKED
CONST	PROCEDURE
DIV	PROGRAM
DO	REAL
DOWNTWO	RECORD
ELSE	REPEAT
END	SET
FILE	STRING
FOR	THEN
FORWARD	TO
FUNCTION	TYPE
GOTO	UNTIL
IF	VAR
IN	WHILE
INTEGER	WITH
LABEL	

ADDITIONAL APPLE UCSD PASCAL RESERVED WORDS

EXTERNAL
SEGMENT
SEPERATE
UNIT
USES
INTERFACE
IMPLEMENTATION

DEFAULT PARAMETERS FOR SYSTEM MISCINFO

Parameter	Default Value
Field Name	
BACKSPACE	left-arrow key (CTRL-H)
EDITOR ACCEPT KEY	CTRL-C
EDITOR ESCAPE KEY	ESC
ERASE LINE	NUL (ASCII 0)
ERASE SCREEN	CTRL-L
ERASE TO END OF LINE	CTRL-J
ERASE TO END OF SCREEN	CTRL-K
HAS 8510A	FALSE
HAS BYTE FLIPPED MACHINE	FALSE
HAS CLOCK	FALSE
HAS LOWER CASE	FALSE
HAS RANDOM CURSOR ADDRESSING	TRUE
HAS SLOW TERMINAL	FALSE
KEY FOR BREAK	NUL (ASCII 0)
KEY FOR FLUSH	CTRL-F
KEY FOR STOP	CTRL-S
KEY TO DELETE CHARACTER	left-arrow key (CTRL-H)
KEY TO DELETE LINE	CTRL-X
KEY TO END FILE	CTRL-C
KEY TO MOVE CURSOR DOWN	CTRL-L
KEY TO MOVE CURSOR LEFT	left-arrow key (CTRL-H)
KEY TO MOVE CURSOR RIGHT	right-arrow key (CTRL-U)
KEY TO MOVE CURSOR UP	CTRL-O
LEAD IN FROM KEYBOARD	NUL (ASCII 0)
LEAD IN TO SCREEN	NUL (ASCII 0)
MOVE CURSOR HOME	CTRL-Y
MOVE CURSOR RIGHT	CTRL-
MOVE CURSOR UP	CTRL-
NON PRINTING CHARACTER	?
PREFIXED [DELETE CHARACTER]	FALSE
PREFIXED [EDITOR ACCEPT KEY]	FALSE
PREFIXED [EDITOR ESCAPE KEY]	FALSE
PREFIXED [ERASE LINE]	FALSE
PREFIXED [ERASE SCREEN]	FALSE
PREFIXED [ERASE TO END OF LINE]	FALSE
PREFIXED [ERASE TO END OF SCREEN]	FALSE
PREFIXED [KEY FOR BREAK]	FALSE
PREFIXED [KEY FOR FLUSH]	FALSE
PREFIXED [KEY TO MOVE CURSOR DOWN]	FALSE
PREFIXED [KEY TO MOVE CURSOR LEFT]	FALSE
PREFIXED [KEY TO MOVE CURSOR RIGHT]	FALSE

PREFIXED [KEY TO MOVE CURSOR UP]	FALSE
PREFIXED [KEY FOR STOP]	FALSE
PREFIXED [KEY TO DELETE CHARACTER]	FALSE
PREFIXED [KEY TO DELETE LINE]	FALSE
PREFIXED [KEY TO END FILE]	FALSE
PREFIXED [MOVE CURSOR HOME]	FALSE
PREFIXED [MOVE CURSOR RIGHT]	FALSE
PREFIXED [MOVE CURSOR UP]	FALSE
PREFIXED [NON PRINTING CHARACTER]	FALSE
SCREEN HEIGHT	24
SCREEN WIDTH	79LSE
STUDENT	FALSE
VERTICAL MOVE DELAY	0

FORMAT AND DISK COPY COMMANDS

- | Disk Drive # | s Pascal Drive
Number | Slot | Drive |
|--------------|--------------------------|------|-------|
| | #4: | 6 | 1 |
| | #5: | 6 | 2 |
| | #9: | 7 | 1 |
| | #12: | 7 | 2 |
- To Boot on 2 Drive System
 - APPLE 1: in Drive #4:
 - APPLE 2: in Drive #5:
- To see right 40 characters of screen type CTRL-A
- # To format a disk after booting
 - to "command" line type
 - to "execute" what file type Apple 3: Formatter
 - put Apple 3: Disk in drive #5:
 - hit return
 - type 5 or (drive # you put disk to be formatted in)
 - after formatting hit return & remove formatted disk
- To copy a disk
 - after formatting a blank disk put it in drive #4: Type F for the filer
 - select filer command T
 - to transfer your disk to the blank disk answer
 - TRANSFER? your disk name or drive number (ie. #4:)
 - TO WHERE? blank: or drive number
 - TRANSFER 280 BLOCKS? (Y/N) Y for full disk
 - DESTROY BLANK? Y
- To display disk contents use F for filer & type L
 - DIR LISTING OF? disk name or drive #

P-MACHINE OP-CODES

Dec	Hex	Mnemonic	Dec	Hex	Mnemonic	Dec	Hex	Mnemonic
0	00	SLDC	0					
1	01	SLDC	1					
.	.	.	.					
126	7E	SLDC	126			213	D5	BPT
127	7F	SLDC	127			214	D6	XIT
128	80	ABI				215	D7	NOP
129	81	ABR	172	AC	XJP	216	D8	SLDL 1
130	82	ADI	173	AD	RNP	217	D9	SLDL 2
131	83	ADR	174	AE	CIP	218	DA	SLDL 3
132	84	AND	175	AF	EQU	219	DB	SLDL 4
133	85	DIF	176	B0	GEQ	220	DC	SLDL 5
134	86	DVI	177	B1	GRT	221	DD	SLDL 6
135	87	DVR	178	B2	LDA	222	DE	SLDL 7
136	88	CHK	179	B3	LDC	223	DF	SLDL 8
137	89	FLO	180	B4	LEQ	224	E0	SLDL 9
138	8A	FLT	181	B5	LES	225	E1	SLDL 10
139	8B	INN	182	B6	LOD	226	E2	SLDL 11
140	8C	INT	183	B7	NEQ	227	E3	SLDL 12
141	8D	IOR	184	B8	STR	228	E4	SLDL 13
142	8E	MOD	185	B9	UJP	229	E5	SLDL 14
143	8F	MPI	186	BA	LDP	230	E6	SLDL 15
144	90	MPR	187	BB	STP	231	E7	SLDL 16
145	91	NGI	188	BC	LDM	232	E8	SLDO1
146	92	NGR	189	BD	STM	233	E9	SLDO2
147	93	NOT	190	BE	LDB	234	EA	SLDO3
148	94	SRS	191	BF	STB	235	EB	SLDO4
149	95	SBI	192	C0	IXP	236	EC	SLDO5
150	96	SBR	193	C1	RBP	237	ED	SLDO6
151	97	SGS	194	C2	CBP	238	EE	SLDO7
152	98	SQI	195	C3	EQUI	239	EF	SLDO8
153	99	SQR	196	C4	GEQI	240	F0	SLDO9
154	9A	STO	197	C5	GRTI	241	F1	SLDO10
155	9B	IXS	198	C6	LLA	242	F2	SLDO11
156	9C	UNI	199	C7	LDCI	243	F3	SLDO12
157	9D		200	C8	LEQI	244	F4	SLDO13
158	9E	CSP	201	C9	LESI	245	F5	SLDO14
159	9F	LDCN	202	CA	LDL	246	F6	SLDO15
160	A0	ADJ	203	CB	NEQI	247	F7	SLDO16
161	A1	FJP	204	CC	STL	248	F8	SIND 0
162	A2	INC	205	CD	CXP	249	F9	SIND 1
163	A3	IND	206	CE	CLP	250	FA	SIND 2
164	A4	IXA	207	CF	CGP	251	FB	SIND 3
165	A5	LAO	208	D0	LPA	252	FC	SIND 4
166	A6	LSA	209	D1		253	FD	SIND 5
167	A7		210	D2		254	FE	SIND 6
168	A8	MOV	211	D3	EFJ	255	FF	SIND 7
169	A9	LDO	212	D4	NFJ			

Pascal Fast Floating-Point Processor Intrinsic Routines

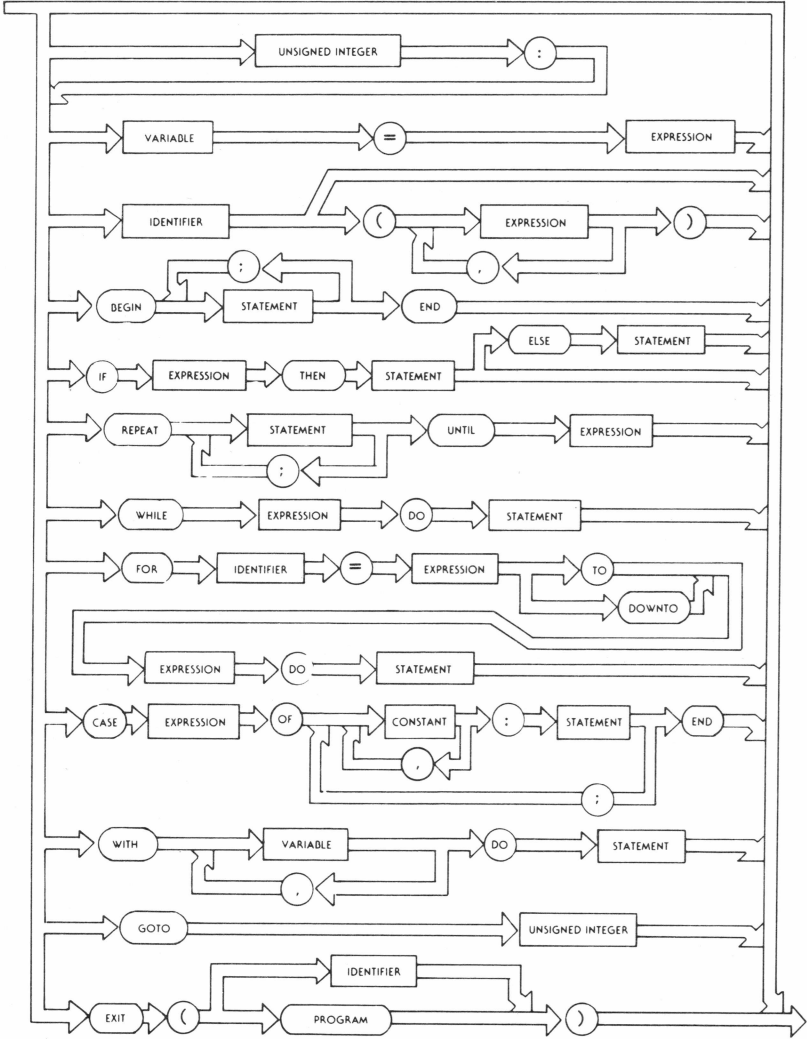
(Requires Computer Station's Pascal Fast Floating Point Board)

TRANSAM function list

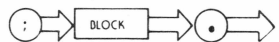
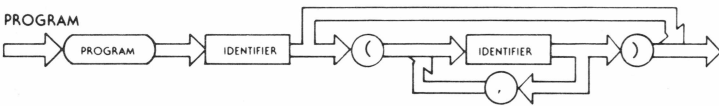
To use these functions you must have 'USES TRANSAM' after PROGRAM statement.

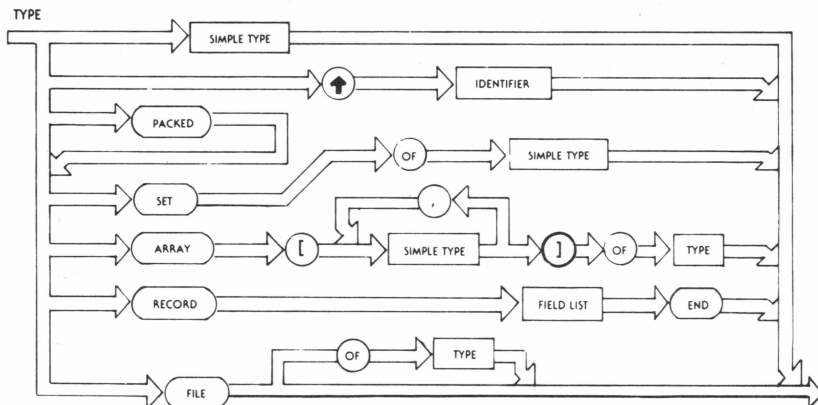
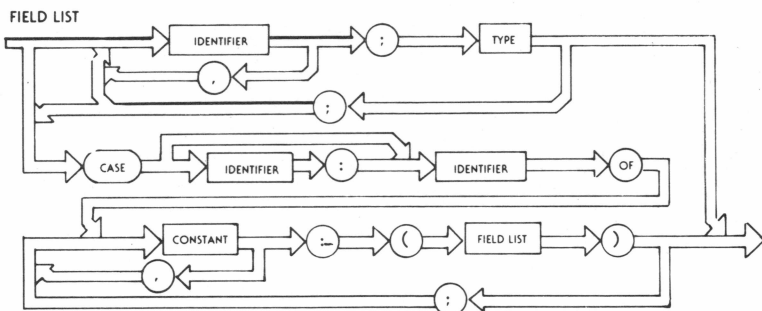
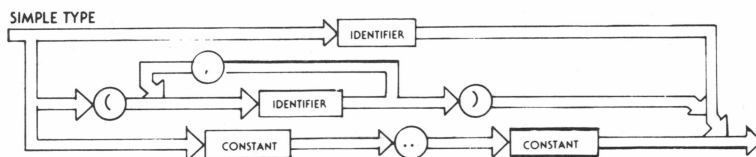
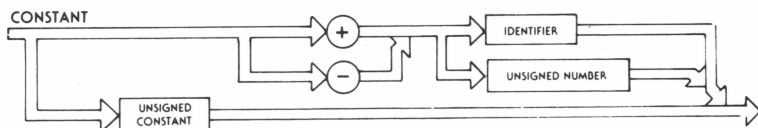
SIN(R)	Returns the value of the Sine of R (in radians). Result: Real
COS(R)	Returns the value of the Cosine of R. Result: Real
TAN(R)	Returns the value of the Tangent of R. Result: Real
ASIN(R)	Returns the value of the ArcSine of R. Result: Real
ACOS(R)	Returns the value of the ArcCosine of R. Result: Real
ATAN(R)	Returns the value of the ArcTangent of R. Result: Real
SINH(R)	Returns the value of the Hyperbolic Sine of R. Result: Real
COSH(R)	Returns the value of the Hyperbolic Cosine of R. Result: Real
TANH(R)	Returns the value of the Hyperbolic Tangent of R. Result: Real
LOG(R)	Returns the value of the base 10 logarithm of R. Result: Real
LN(R)	Returns the value of the natural logarithm of R. Result: Real
EXP(R)	Returns the value of 'e' raised to the power R. Result: Real

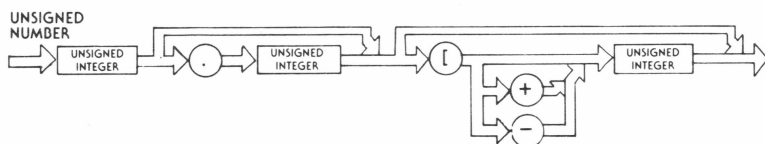
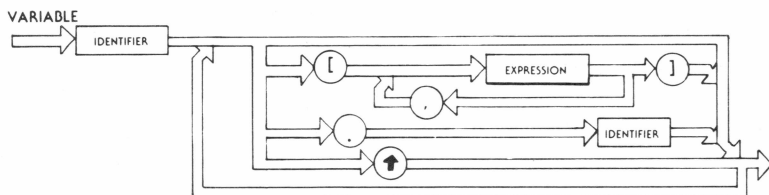
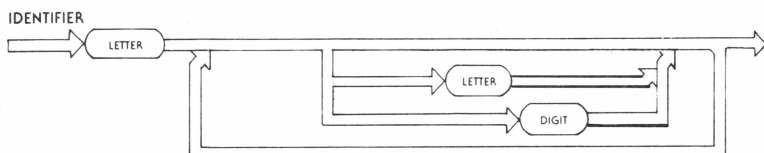
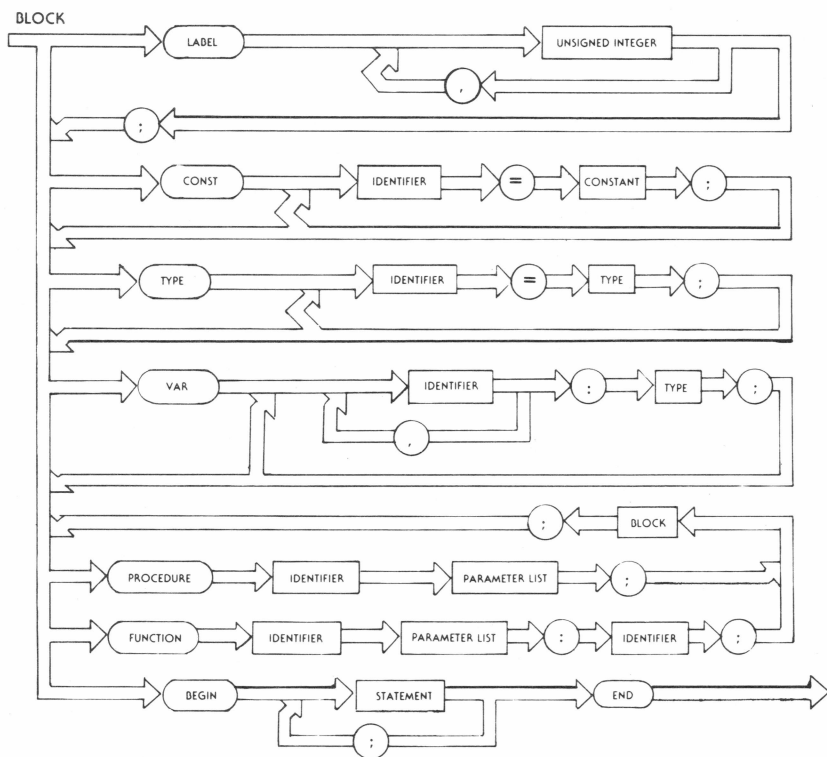
STATEMENT

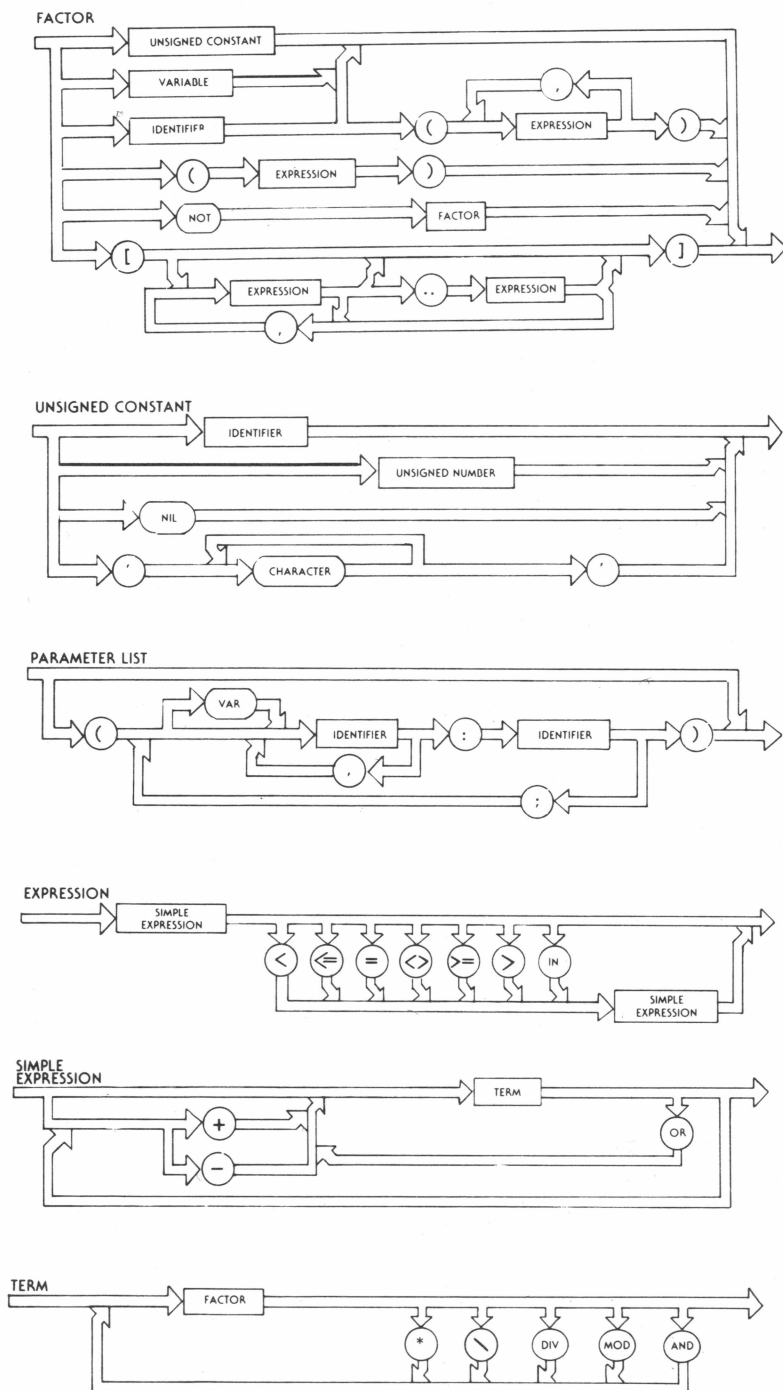


PROGRAM



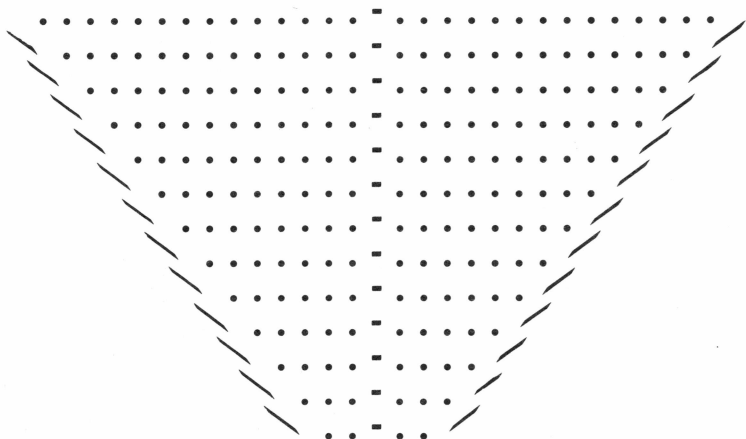






MACRO SCED

[] M A C R O - S C E D ! []



X

WRITTEN BY DAVID CHI
©COPYRIGHT 1980
BY
COMPUTER STATIONS, INC.

MACRO-SCREEN EDITOR

MACRO-SCED is a cursor oriented screen editing tool for the novice as well as for the serious programmer. In fact, any one who types in any program at all, whether copying from books or writing his own, cannot afford to be without it. It's a great time saver. Editing BASIC programs actually becomes fun. MACRO-SCED is easy to use and yet it has features found previously only on expensive terminals.

Commands are executed through CONTROL characters. It works in Integer Basic or Applesoft (ROM or RAM) and in monitor. This machine language program is completely transparent to the user and only uses 1.5K of memory. Basic activation programs and a macro building program are provided.

MACRO-SCED EDITOR

TABLE OF CONTENTS

SUMMARY OF FEATURES	3
ACTIVATING THE SCREEN EDITOR	3
SUMMARY OF COMMANDS	5
OUTPUT CONTROLS	6
THE KEYBOARD MACRO FUNCTION	8
[A] Generating a Macro	9
Prompt charactrs and meaning	9
Memory usage	9
Zero page usage	9
Page 2 usage	9
Summary of data locations use by macro-sced	10
Format of macro-tables	10
Format of macro-entry	10
To start a new table	11
Standard macro table	11
[B] Special Macros & System Macros	12

SUMMARY OF FEATURES

The Macro-Screen Editor is packed with convenient and powerful features. Following is a quick overview. There are four main groups of functions:

Cursor Editing Commands —

Easy cursor movement controls including cursor jumping.

Cursor homing and position saving.

Character or Line insertion and deletion.

Copy an entire screen-line with a single command.

Automatic line copying with user adjustable speed.

Plus other editing aids, such as displaying the input buffer before hitting 'RETURN', steady inverse block cursor to save eye strain.

Output Controls —

Complete screen output (listing) controls including stop-, slow-, page-, and end-list.

Macro Functions —

User definable keyboard Macros. Two key sequence to type or execute often used Basic/DOS commands, statements or phrases. (up to 56 macros in two tables).

Multiple tables possible, each pair selectable by a single POKE command.

Complex macros: All MACRO-SCED commands can be included.
Manual and Auto-chaining to another macro.

Macro defining programs provided.

List all macros defined at will.

Generates 3 characters not available on the keyboard:

- back slash, under-line, and the left-hand bracket.

Screen-Save function —

Special screen contents saving feature, up to 35 screens limited only by free memory.

Screens saved remain intact from program to program, or from one Basic to another.

Use screen-save feature to display a summary of commands, screen buffer directory and other information.

Screen-writer and screen dump to printer programs provided.

Compatible with the Autostart ROM or APPLE II Plus.

ACTIVATING THE SCREEN-EDITOR

Starting up

The Screen-Editor diskette includes a boot up program that makes the activation process very simple. There are two ways to start up. One is to boot with the editor diskette. If you have the Autostart ROM, the disk

will boot as soon as you turn the computer on. If you have the original monitor ROM, boot the disk as usual by typing 'ó(CTRL-P) CR'.

The other way is to 'RUN' the HELLO program if you have already booted up from another diskette. The HELLO program is in Applesoft but there is another boot up program written in Integer Basic. You can use either one to start up. If you wish to boot up in Integer Basic, rename HELLO,FP HELLO and rename INT HELLO, HELLO.

If you do not have ROM Applesoft, save a copy of the Integer program "Applesoft", without running it, on the diskette. You will need it to run Macro-Builder.

Reentry

The booting program will display reentry information while it is loading the screen-editor. This reentry is sometimes referred to as a "warm start". Reactivating Macro-Sced while it is active is fine. In Applesoft use 'G' <CR> to do a warm start. This means that the ampersand entry location at hex address \$3F5 must be unchanged since starting up the editor. If the contents have been changed, use 'CALL 960' instead. To reenter from monitor, type 'CTRL-Y <CR>' or '3COG <CR>'. Location 960 is hex address \$3C0. If you are in Integer Basic, also use 'CALL 960'. The warm start pointers for 'G' and 'CTRL-Y' are set up by the start-up program and can be restored by running the HELLO program again if destroyed. If the system hangs after doing a warm start, hit 'RESET' and reboot. The editor will not work correctly if memory locations 960-1023 (\$3C0-3FF) have been destroyed. See Appendix (B) on system memory usage for more details.

Warm-Start Summary:

From Integer Basic	——— CALL 960 <CR>
From Applesoft	——— 'G' <CR> or CALL 960 <CR>
From Monitor	——— CTRL-Y <CR> or 3COG <CR>

The Screen-Editor resides just below DOS and above the DOS buffers. Thus it becomes part of DOS and is practically indestructable until DOS itself is destroyed. It is unaffected by 'INT', 'FP' or 'MAXFILES' commands!

Functionally the editor intercepts all keyboard inputs and screens them for Macro-Sced commands before passing them on the DOS. Just like DOS it will be disconnected if the 'RESET' key is hit. If you have the Autostart ROM, you are protected from accidental disconnections. The screen-editor will also be disconnected if 'PR#0,IN#0' have been executed from keyboard or from within a Basic program (as in programs using a printer). It is also possible to disconnect the output controls (e.g. stop-list function) only, by a 'PR#0' and not affect the input controls (e.g. cursor movements) or vice versa. In any case, do a warm start to reconnect both types of controls. Re-running the HELLO program will also work but usually not necessary.

Running BASIC programs

In most cases, Macro-Sced will not affect the running of a Basic program. As a matter of fact, it might even be advantageous as the section on

command descriptions will point out. But if the program uses control-keys in its input, you will want to disconnect the editor before running the program. You can do so by pressing 'ESC' and then hitting 'RETURN'. This is the same as typing 'PR#0' and 'IN#0'. Remember to re-activate Macro-Sced afterwards. You will not want to program without it.

You can tell if Macro-Sced is activated by pressing 'ESC' twice. Press it once to reveal a flashing '^' (up-arrow) prompt, indicating that the editor is reading your command. Press it again to return to the normal cursor. If you are using the non-flashing cursor, it will be obvious whether Macro-Sced is active.

SUMMARY OF COMMANDS

CURSOR EDITING COMMANDS

Cursor Moving —

CTRL-W	Move cursor up one line
CTRL-A	Move cursor left one space
CTRL-S	Move cursor right one space
CTRL-Z	Move cursor down one line
ESC, CTRL- W,A,S,Z	Move cursor U,L,R,D 8 lines or spaces

Cursor Jumping —

CTRL-T	Tab cursor to next non-space character
CTRL-R	Tab cursor back to previous non-space character (Reverse Tab)
CTRL-K	Jump cursor to next occurrence of the character at the present cursor position
CTRL-J	Jump cursor back to previous occurrence of the character at the present cursor position

Cursor position —

CTRL-SHIFT-P	Home cursor to upper-left corner of screen
CTRL-B	Bottom home cursor to lower-left corner of screen
CTRL-Q,S	Save the present cursor position
CTRL-Q,R	Restore cursor back to last position saved

Input —

CTRL-Y	Copy to last non-space character on screen line
CTRL-L	Start copying all characters until a key is hit

The copy rate is changed with a single 'POKE' as follows:

Hit 'ESC', then 'CTRL-L'

This is actually a Macro function. Note that 'CTRL-L' is used both as command and macro keys! The default value is 160 which is about four times faster than using the '→' and 'REPT' combination to copy characters. To enter another value, delete '160' with the '<' key, enter the desired number and hit 'CR'. The valid range is from 1 (slowest) to 255 (fastest!).

The useful range is from about 100 to 200. The following chart shows the approximate relative speeds:

	Speed Value	Char./Second	Times Faster Than 'REPT'
	100	15	1.5 X
	120	20	2 X
	140	30	3 X
(default)	160	40	4 X
	180	60	6 X
	200	120	12 X

CTRL-I	Insert a line. Split line at cursor and scroll up
CTRL-O	Insert a character at cursor position
CTRL-N	Delete a character at cursor position
CTRL-V	Inverse block cursor ON and OFF
CTRL-SHIFT-N	Display the input buffer thus far entered

Erase —

CTRL-E	Clear from cursor to end of screen line
CTRL-F	Clear from cursor to end of screen page

OUTPUT CONTROLS

Any key	Stops listing or character output
Any key after stop	Resumes listing
RETURN after stop	Ends listing -
CTRL-P	Page-list mode ON and OFF

Slow list controlled by a 'POKE' Macro command

MACRO FUNCTION

ESC, key	Generates a Macro
ESC, RETURN	Deactivate MACRO-SCED and exit
CTRL-SHIFT-M, M	Display macro tables

SCREEN-SAVE FUNCTION

CTRL-SHIFT-M, S, #	Save screen contents into memory buffer #
CTRL-SHIFT-M, L, #	Load screen with contents of buffer #
CTRL-SHIFT-M, B	Display usable range of screen buffers

OUTPUT CONTROLS

The following commands will not only control the listing of a program but also control the output generated by a Basic program.

(1) Stop-List, End-List

When a program is being listed, hitting any key will immediately halt the listing. Hitting any key again will allow the listing to resume. Hitting the

'RETURN' key after a halt will terminate the listing. The space bar is a convenient key to use for this control.

(2) Slow-List

The speed of a listing or any program output to the screen can be controlled by changing the value at a special location using a 'POKE' statement or command. A value of 1 is slowest and 255 is fastest. The default value is 255. To change the output rate type:

ESC, then CTRL-R > POKE 969,255

Use the '<' key to delete the default value and enter any number from 1 to 255 for slower listing. Just hit 'RETURN' to reset maximum speed.

"POKE 696,

(1 to 255)" for slower output

"POKE 969,255 for fastest output

(3) Page-List mode

CTRL-P Page list mode ON/OFF

An inverse 'P' indicates that the page-list mode has been turned ON. A line feed with no letter indicates page-list mode is turned OFF.

In the ON mode, the listing will pause with a cursor prompt after every 22 screen lines. Hitting a key will list the next page. Displaying 22 lines per page will usually allow an entire program line at the bottom of the screen to be viewed when advanced to the next screen page. hitting the 'RETURN' key will terminate the listing but the page-list mode is still ON. Hitting CTRL-P at a pause will both end the listing and turn the page-list mode OFF at the same time.

The Stop-List and Slow-List command will also work while the Page-List mode is on and can be used in combination for full listing control.

To turn the Page-List mode ON and OFF from within a program, enter the following statements:

"POKE 48,22 Set no. of lines per page

Then: "POKE970,PEEK(970)+128" turns the mode ON

"POKE970,PEEK(970)-128" turns the mode OFF

Turning the mode ON after it is ON already will result in a range error. You can test the mode by PEEKing at 970.

"PEEK(970) >= 128" Page-List mode ON

"PEEK(970) < 128" Page-List mode is OFF

(4) End-List

The following applies to both Stop-List and Page-List commands

The End-List command (hitting 'RETURN') should not be used from within a program because it reenters Basic through a jump to location \$3DO. Thus ending an output will also end the Basic program itself as well.

If you are using a printer and executed a 'PR#O' is executed from within a program or keyboard. However the other cursor commands remain functional.

To reactivate the listing controls again, do warm start with 'CALL 960' from Integer Basic or 'G' from Applesoft. Doing this from within a program will also end the program.

THE KEYBOARD-MACRO FUNCTION

In the process of coding a program, there are many Basic commands such as 'PRINT' or 'LIST' that are used over and over again. So are certain DOS commands such as 'CATALOG'. The Keyboard-Macro function relieves the programmer from typing in every character of all these commands and it does more.

A Keyboard-Macro (KBD-Macro or Macro for short) is a string of characters that can be 'typed' by simply pressing a two-key sequence. The effect is as though each of the characters has been typed on the keyboard itself. Thus anything that can be typed on the keyboard can be incorporated into a macro. This includes commands, statements, expressions, a whole line of Basic, etc. . . .

(A) Generating a KBD-Macro

The strings to be used for the macros are stored in the Macro-table. Each macro is associated with a key on the keyboard. The Editor comes with a pre-defined set of macros. They can also be user defined and we'll get into that a little later.

The 'ESC' key is the command key for generating macros: Press 'ESC', then a 'key'.

There are two tables of macros. The 'key' to be pressed depends on which table the macro is stored in. Table 1 is accessed by a 'CTRL' key. Table 2 is accessed by an alphabetic key.

Thus to generate a macro from table 1: Press 'ESC', then a key from 'CTRL-A' through 'CTRL-Z' (plus 'CTRL-SHIFT-M' and 'CTRL-SHIFT-N'). e.g. Hitting 'ESC', then 'CTRL-Z' will move the cursor down 8 lines.

To generate a macro from table 2: Type 'ESC', then a key from 'A' through 'Z' (plus 'SHIFT-M' and 'SHIFT-N'). e.g. hitting 'ESC', then 'C' will print out the disk catalog.

The macro mode will be exited after the macro is completed.

The macro command will be cancelled if an invalid key is pressed. Pressing another 'ESC' will also cancel the macro command. If a macro has not yet been defined, nothing will be generated, of course.

NOTES:

'ESC-SHIFT-P' has been retained for clearing the screen and homing the cursor.

'CTRL-M' is the same as the 'RETURN' key. This macro has been defined to exit Macro-Sced.

'CTRL-H' is the same as the left-arrow key and 'CTRL-U' is the same as the right-arrow key.

PROMPT CHARACTERS AND MEANING

Inverse Block Cursor	Same as regular cursor
Inverse '+'	Save/Restore cursor position
Inverse '?'	Waiting for for screen-save command or Macro listing command
Inverse '#'	Waiting for screen-buffer no.
Flashing ' '	Waiting on Macro key
Inverse 'P'	Page-list mode turned on

MEMORY USAGE

The Macro-Sced is written entirely in machine language and the editor itself is only 1K long. Its compact size is due to the fact that it uses the monitor subroutines extensively. Only subroutines that are common to both the original and Autostart monitor are used. The macro-tables alone are .5K long. The editor physically becomes part of DOS when loaded and is protected from accidental erasure.

Macro Screen-Editor and Macro-Tables address:

System Size:	32K	48K	Length to the end
Macro-Sced:	22272 (\$5700)	3865G (\$9700)	1536 (\$600)
Location			:
Table 1	: 23296 (\$5B00)	39680 (\$9B00)	: 512 (\$200)
Table 2	: 23552 (15C00)	39936 (\$9C00)	: : 256 (\$100)

Use the following DOS format:

"BLOAD (filename), A\$(hex address)" <CR>

"BSAVE (filename), A\$(hex address), L\$(hex length)" <CR>

The ideal location for Macro-Sced is between DOS and its buffers. But if for good reasons it cannot be at this location, the Screen-Editor can be loaded elsewhere because it is relocatable! It can be BRUN anywhere in free memory as long as it starts at the beginning of a memory page. In this case, the documented address in the table above will not apply and warm start pointers are not set-up. This set-up should only be used by those who are familiar with memory allocation.

ZERO PAGE USAGE

60-69 (\$3C-\$45) temporary storage by certain subroutines. These locations need not be specially preserved by the user.

PAGE 2 USAGE

Macro-Patcher uses part of page 2 which is the input buffer for storage of the message strings. Addresses used are \$2C0-\$2FF.

SUMMARY OF DATA LOCATIONS USE BY MACRO-SCED

Location (Hex)	Usage in Screen-Editor
960 (\$3C0)	CALL here for warm start.
961 (\$3C1)	POKE with zero & CALL for cold start.
962 (\$3C2)	Memory page where Macro-Sced, resides. Address = PEEK (962) * 256.
963- (\$3C3- 968 3C8)	Internal use only
969 (\$3C9)	Character output speed. 1 = slowest, 255 = fastest (default) (use macro 'CTRL-R')
970 (\$3CA)	Page-list and inverse cursor flags. Default = 0 (OFF) PEEK (970) + 128 = Page-list ON (also need POKE 48,22) PEEK (970) - 128 = Page list OFF PEEK (970) + 64 = Inverse cursor ON PEEK (970) - 64 = Inverse cursor OFF
971 (\$3CB)	Auto-line copy rate. Default = 160 1 = slowest, 255 = fastest
972 (\$3CC)	Horizontal cursor position saved (0-39).
973 (\$3CD)	Vertical cursor position saved (0-23).
974 (\$3CE)	Memory page of Macro-Table 1. Address = PEEK (974) * 256
975 (\$3CF)	Supress range check flag for screen-buffers. 0 = check (default), 255 = supress check (use macro 'CTRL-SHIFT-M')

FORMAT OF MACRO-TABLES

Each table is 256 bytes (1 memory page, \$100 hex) long. Macro entries are stored sequentially in each table. Each entry consist of a macro-key code and the string macro itself. The macro-key code is a number from 1 to 30 corresponding to A through SHIFT-N in as ASCII table. The next macro-key code marks the end of the previous macro. The end of the table is indicated by a zero (byte #\$00). The two tables are identical in its macro-key code. The choice of which table to use is determined in Macro-Sced's subroutine itself. The macro-key codes need not be in numeric sequence. The macro generating routine will search through the entire table for a macro requested.

FORMAT OF A MACRO ENTRY

Type:	Standard	Manual-chain	Auto-chain
Macro-Key:	1-30 (\$01-1D)	1-30 (\$01-1D)	1-30 (\$01-1D)
Macro- :	ASCII Characters (Bit 7 set) including CTRL-Characters		

String	:			
End of	:	Next	'ESC'	Not on KBD
Macro	:	key-code	155 (\$9B)	254 (\$FE)
Next Byte	:	Macro String	Next	Key-code to
			key-code	chain to
End of Table:		00	00	00

TO START A NEW TABLE

- (1) Activate Macro-Sced
- (2) LOAD Macro Builder
- (3) POKE Table 1 address, 0 or POKE Table 2 address, 0
- (4) RUN

Refer to table in Appendix (B) for Macro-table addresses. They can also be derived from the following expression:

Table 1 address = PEEK(974) * 256

Table 2 address = PEEK(974) * 256 + 256

STANDARD MACRO TABLE

[= ESC.

> = CHAIN TO NEXT MACRO

H = '←' DEL.

U = '→' COPY M = RET.

MACRO-TABLE 1 (236 BYTES LONG)

CTRL-A = "AAAAAAAAA["
 CTRL-B = " "
 CTRL-C = "CHR\$("
 CTRL-D = "MASTER DISKETTE"
 CTRL-E = "APPLE]["
 CTRL-F = "FLASH"
 CTRL-G = "
 CTRL-H = "\"
 CTRL-I = "INVERSE"
 CTRL-J = "CALL768M"
 CTRL-K = "
 CTRL-L = "COPY RATEXPOKE
 971, 160
 CTRL-M = "PR#OMIN#OM"
 CTRL-N = "NORMAL"
 CTRL-O = "
 CTRL-P = "?TAB["
 CTRL-Q = "
 CTRL-R = "SPEEDXPOKE
 969, 255"
 CTRL-S = "SSSSSSSS["
 CTRL-T = "
 CTRL-U = " ← "

MACRO-TABLE 2 (245 BYTES LONG)

A = "MMACRO-SCED READYGM"
 B = "3DOGM"
 C = "CATALOGM"
 D = "DATA"
 E = "CALL-936M W"
 F = "?FRE(0)M"
 G = "GOTO"
 H = "HTAB"
 I = "INPUT"
 J = "=RESETXCAL-167"
 K = "PEEK["
 L = "LIST"
 M = "CALL-151M"
 N = "NEXT"
 O = "POKE"
 P = "PRINT"
 Q = "
 R = "RETURN"
 S = "GOSUB"
 T = "THEN"
 U = "UNLOCK"

CTRL-V = "VAL["	V = "VTAB4MAZ"
CTRL-W = "WWWWWWWW["	W = "POKE33,33M→L"
CTRL-X = "DELETE"	X = "TEXT"
CTRL-Y = "LOAD YM"	Y = "-16368, 0"
CTRL-Z = "ZZZZZZZ["	Z = "16384)"
CTRL-SHIFT-M =	SHIFT-M = "["
"<BUFF>XPOKE 975, 0"	
CTRL-SHIFT-N = "TABLE@XPOKE	SHIFT-N = "PRINT (PEEK (203)
974, PEEK (962) + 4"	-PEEK (205) + 2) / 4;" K FREE "M"
HIT A KEY TO NEXT TABLE	

An advantage of the auto-chain macros over the multi-functional macros is that we are not limited to executing the whole "chain" but are able to enter the "chain" in the middle. Thus some macros can be designed to be part of several "chains".

EX.7: Macro 'E' automatically chains to macro 'W' which in turn chains to macro 'L'.

'ESC,W' will set a narrow window and go to macro 'L'

'ESC,L' is the LIST command.

Thus macro 'L' is used in three different ways.

(B) Special Macros and System Macros

The two macro tables come with a pre-defined set of useful macros but the user is free to redefine or add to them. There are three special macros that generate three keys that are not available on the keyboard. The macros are:

ESC, '<'	for the back-slash
ESC, '>'	for the under-line
ESC, '{'	for the left-hand bracket

The under-line character is not accepted by the Integer Basic interpreter and therefore it cannot be used in a quoted string.

Macro 'A' is reserved for the greeting message whenever Macro-Sced is activated. It is not a valid command. The 'CR' is for skipping a line only.

Macro 'CTRL-Q' is deliberately not defined so as to be used for exiting (quiting) the macro mode during cursor jumping.

There are also several macros that are used to 'POKE' different locations to control a variety of functions. Some of these system macros have been introduced in their appropriate sections. They are:

ESC, CTRL-R	for character output rate
ESC, CTRL-L	for auto-line copy rate
ESC, CTRL-SHIFT-M	for out-of-range screen buffer check
ESC, CTRL-SHIFT-N	for macro tables location

These special and system macros should not be changed. Note that all these macros, except 'A' and 'SHIFT-M', are in table 1.

UTILITIES

APPLESOFT PROGRAMER'S ASSISTANT

COMMANDS

GAuto <START# > [, <INCREMENT >]

Automatic line numbering starting with start# and incrementing by 10 unless specified by the increment No.

Note: To get line number, press space bar.

GManual Turns off automatic line numbering

GRenumber [<START# >] [, <INCREMENT# >]

Renumbers all lines starting with 100 and with an increment of 10. If a start number is specified, it is the starting line number. If the increment is specified, it is the value added to each successive line number.

GRenumber <START >, <INCREMENT# >, <FIRST# >, <LAST# >

Renumbers only the program area specified by the start # and the last #.

— CAUTION —

If your line numbering will overlap and interleave you will get a warning message and renumbering will abort. Interleaved or duplicate line number.

— SPECIAL NOTE —

However, if your new lines will wholly fit between two line numbers, that program section will be moved to occupy the new line numbers. This can be used to an advantage - but watch it.

GHold — Used to hold an overly program above HIMEM while a new program is loaded.

— CAUTION —

If this program has lines that are the same as the next program, the program in hold will overlay or replace the duplicate lines.

GMerge — After your second program is loaded, merge is used to overlay the program in hold. Lines will be merged together and conflicting line numbers will be replaced by the program in hold.

GCompress — Removes all REM statements

GLength — Returns both decimal and hexadecimal program length.

GShow — Displays all control codes in program in inverse video.

GNo show — Turns off control character display

&Keys — After use, it adds the following characters to the keyboard.

CTRL-O	—	Underscore
CTRL-L	\	Backslash
CTRL-K	[	Left Bracket

NOTE: SHIFT-M is already] right bracket to cancel &Keys use
GMA. &K does not work while auto line number is on.

&Xref — Lists all variables and the line number they are on.

HI-RESOLUTION GRAPHICS

CHARACTER COMMANDS

CONTROL SEQUENCE SUMMARY

CHAR	FUNCTION	OPTION
CTRL-A n	Select Character Set n	CTRL-A Primary is Page 1 *
CTRL-B	Begin Block Display	CTRL-B Primary is Page 2
CTRL-C	Carriage Return	CTRL-C Complement
CTRL-D	Delimit Block Display	CTRL-D Display Primary
CTRL-E	Clear to End of Line	
CTRL-F	Clear to End of Screen	
CTRL-G		
CTRL-H		
CTRL-I	Inverse Video	
CTRL-J		
CTRL-K	Caps Lock *	
CTRL-L	Lower Case	
CTRL-M		
CTRL-N	Normal Video *	
CTRL-O	Option Select	CTRL-O Overstrike
CTRL-P	Clear Page	CTRL-P Print *
CTRL-Q	Home Cursor	
CTRL-R		CTRL-R Reverse Overlay
CTRL-S	Shift	CTRL-S Scroll *
CTRL-T		CTRL-T Transparent Overlay
CTRL-U		
CTRL-V	Set Text Window (upper-left)	
CTRL-W	Set Text Window (lower-right)	CTRL-W Wrap
CTRL-X		
CTRL-Y	Set Text Window (full screen) *	CTRL-Y Call User Sub A
CTRL-Z	Restore Default Parameters	CTRL-Z Call User Sub B

*Default Parameter

The default Character Set is ASCII-SET

SUMMARY OF ANIMATRIX COMMANDS

STRING INPUT COMMANDS

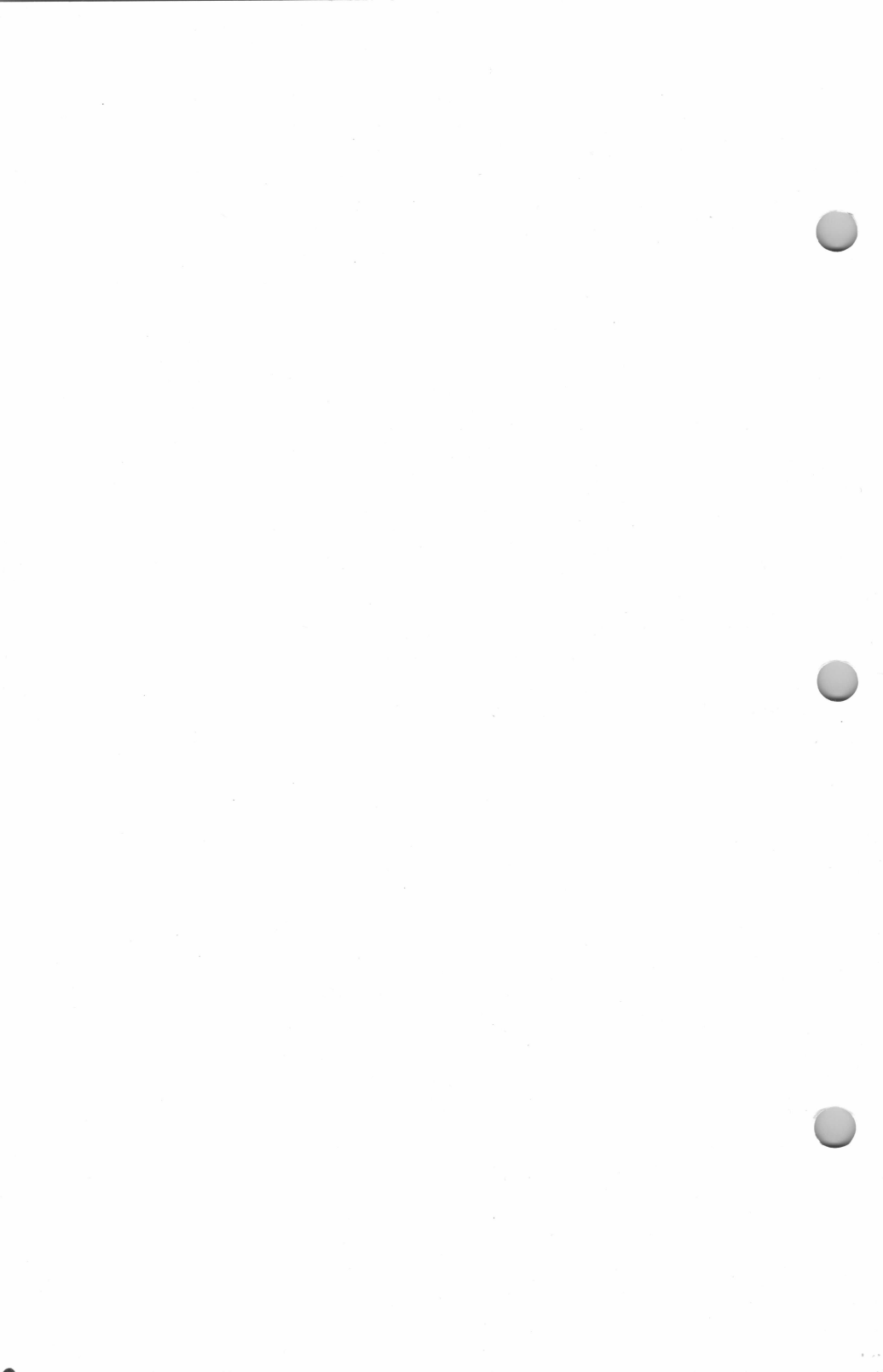
←	move cursor left
→	move cursor right
RETURN	go to next line
CTRL-A	shift to upper case
CTRL-K	left bracket, [
CTRL-L	backslash, \
CTRL-O	underscore, ____
CTRL-Z	shift to lower case

CHARACTER EDITING COMMANDS

Paddle Ø	move grid cursor horizontally
Paddle 1	move grid cursor vertically
Button Ø	turn bit off
Button 1	turn bit on
CTRL-T	toggle high bit for half-dot shift
CTRL-S	same character image

OTHER COMMANDS

ESC	save character set and quit
CTRL-D	display entire character set
CTRL-I	display summary of instructions



APPLE WRITER

MODES (cursor shape indicates mode)

CURSOR	MODE	ENTER FROM TEXT MODE	EXIT → TO GET
☐	Text Mode	(DEFAULT)	[ESC] ^
†	Cursor Control Mode	[ESC] [ESC]	[SPACE] ☐
^	Upper Case Mode	[ESC]	[ESC] [SPACE] ☐
%	Case Change Mode	[ESC] [ESC] CTRL-C	[ESC] † [SPACE] ☐

☐ **TEXT MODE** — Use to enter text in lower case

† CURSOR CONTROL MODE

Key	Cursor Movement
K	Right one character
J	Left one character
I	Up one line
M	Down one line

^ UPPER CASE MODE

[ESC] prior to character works for next character only

% CHANGE CASE MODE (reverse upper & lower case)

- 1) Type [ESC] [ESC] and move the cursor to location for first change.
- 2) Type CTRL-C & use cursor keys to move cursor to change characters or lines

Key	Change Made
K	Change one character right
J	Change one character left
I	Change one line up
M	Change one line down
[ESC]	To stay in Cursor Control but leave change mode
[ANY CHAR]	Leaves change case mode except I,J,K,M

CHARACTER DELETION

←	Deleter character left
CTRL-W	Delete word left + one space
CTRL-X	Delete left til [RETURN] + delete that [RETURN] Max 256 characters

CHARACTER RETRIEVAL

→	Re-Insert characters deleted Max 256 characters
---	--

BLOCK CURSOR CONTROL

Type	Moves Cursor
CTRL-B	To file beginning
CTRL-E	To file end
CTRL-T	Up 12 lines
CTRL-V	Down 12 lines

Mode Usage			
TEXT	CURSOR	UPPER CASE	CASE CHANGE
☐	↑	^	%
✓		✓	
✓		✓	
✓	✓	✓	✓
✓	✓	✓	✓

CONTROL FUNCTIONS

- CTRL-D [RETURN] — Gives access to DOS. All disk functions (i.e. catalog, delete etc.)
- Q [RETURN] — Returns to Apple Writer from DOS.
- CTRL-F — States remaining available bytes of memory
- CTRL-K — Saves a file segment from a distinctive marker to present cursor position. (recommended marker &G)
- CTRL-I — Retrieve file from disk & insert in text. At the point immediately preceding the cursor and leave cursor at end of inserted text.
- CTRL-P — POP - produces audible click when cursor passes over a [RETURN]. Allows detection of line ending over line wrap.
- CTRL-S — Search & Replace. Search forward toward end of file for delimited string
[/ search string / replacement string /]
M = manual selection of optional replacement
CTRL-R — Does Replacement
A = Automatic Replacement
- CTRL-Y — Move block of text. Saves block of text from present cursor position back toward beginning of file to the specified marker. File is saved to disk under the name 'transfer file'. Destination marker is asked and file is inserted prior to that marker.

EXTRAORDINARY CHARACTERS

Type	To Get Ouput To Printer as	On Monitor Screen Shown As
CTRL-N	{	[
[ESC] CTRL-N	[	⌈
SHIFT M	}	]
[ESC] SHIFT M	]	⌋
SHIFT N	~	^
[ESC] SHIFT N	^	⌘

PRINTER COMMANDS

Command	Printing Set To:
!lm{number}	Left margin to column {number}
!rm{number}	Right margin to column {number}
!tm{number}	Top margin blank lines {quantity}
!bm{number}	Bottom margin blank lines {quantity}
!sp{number}	Number of spaces between lines
!fj	Full justify mode (flush right & left)
!lj	Left justify mode (flush left)
!cj	Center justify mode (centered)
!rj	Right justify mode (flush right)
!np	A new page

APPENDIX D: ASCII CHARACTERS

Here is a table showing the characters you type to produce the characters you want printed.

The following sets of characters don't have an upper case version. They will be printed the same whether you type them in upper or lower case:

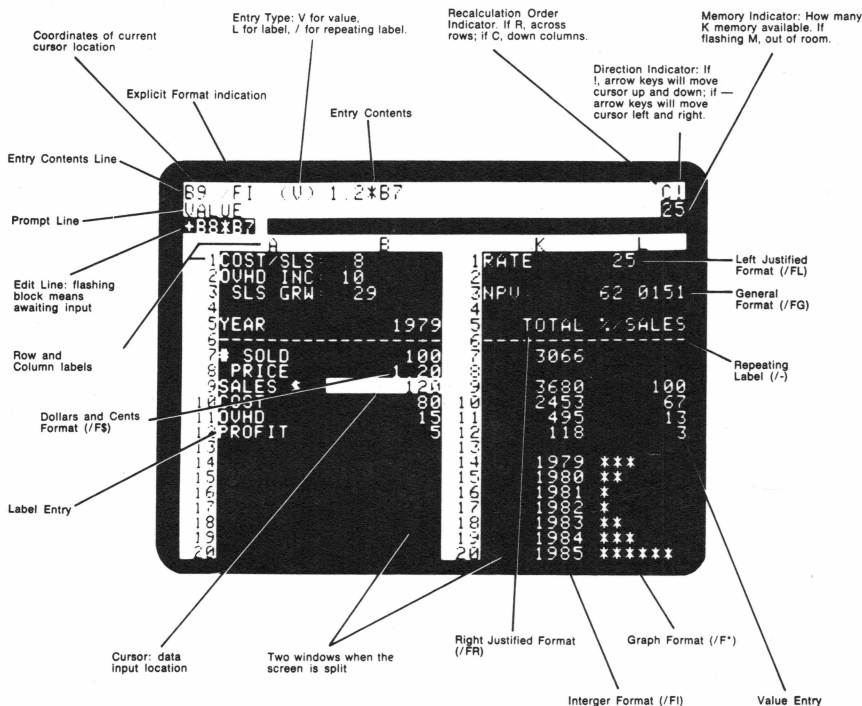
To get this character:	Type this:	To get this character:	Press SHIFT,
------------------------	------------	------------------------	--------------

space	space	!	1
0	0	"	2
1	1	#	3
2	2	\$	4
3	3	%	5
4	4	&	6
5	5	'	7
6	6	(	8
7	7	)	9
8	8	*	:
9	9	+	;
:	:	<	,
'	'	=	_
_	_	>	.
/	/	?	/

The following characters DO have an upper case version, which you get by pressing ESC.

To get this character:	Type this:	To get this character:	Press ESC, then:
a	A	A	A
b	B	B	B
c	C	C	C
d	D	D	D
e	E	E	E
f	F	F	F
g	G	G	G
h	H	H	H
i	I	I	I
j	J	J	J
k	K	K	K
l	L	L	L
m	M	M	M
n	N	N	N
o	O	O	O
p	P	P	P
q	Q	Q	Q
r	R	R	R
s	S	S	S
t	T	T	T
u	U	U	U
v	V	V	V
w	W	W	W
x	X	X	X
y	Y	Y	Y
z	Z	Z	Z
{	CTRL-N	[	CTRL-N
}	shift M	]	SHIFT M
~	shift N	^	SHIFT N
\	shift P	@	SHIFT P

VISICALC™ * REFERENCE CARD



MOVING THE CURSOR

← →

Moves the cursor left, right, up or down.

space bar

Switches the direction indicator between horizontal (—) and vertical (!).

;

If two windows, moves the cursor from one window to the other.

>

Go To command. Type the coordinates to the entry where you want the cursor to go; end with RETURN.

THE ESC KEY

The ESC key is used to recover from simple typing mistakes. It usually erases the last thing that you typed. If you press ESC enough times, it will abort what you are doing and return VisiCalc to a blank prompt line.

VisiCalc computer program and reference card
copyright 1979, 1980 Software Arts, Inc. Reference
card reprinted by permission of Software Arts, Inc.

*VisiCalc™ - Personal Software Inc.

1330 Bordeaux Drive • Sunnyvale, Ca. 94086

SETTING A LABEL ENTRY

Label entries start with a letter (A-Z), or with the quote character ("). Terminate entering a label entry by pressing $\leftarrow$, $\rightarrow$, or RETURN. Correct errors by pressing ESC. The prompt line will say LABEL while a label entry is being typed.

SETTING A VALUE ENTRY

A value entry displays the calculated value of the expression stored at the entry. Expressions consist of numbers, coordinates of other value entries (value references), functions (such as @SUM), arithmetic operators (+ - * / ^) and/or parentheses. Expressions are evaluated strictly from left to right except as modified by parentheses. You must start an expression with a +, a digit (0-9), or one of the symbols @ - (, or #. The prompt line will say VALUE while an expression is being typed. Terminate entering an expression by pressing $\leftarrow$, $\rightarrow$, or RETURN. Errors can be corrected by pressing the ESC key. Examples of expressions are:

12.34	A normal number
.1234E2	A number in scientific notation
2 + 2	An arithmetic expression
+B4	A value reference
2*B4	An expression with a value reference
2*(3 + 4)	An expression with parentheses

If you press ! while entering an expression, VisiCalc will calculate the value of the expression so far and replace the expression on the edit line with the number which results from the calculation.

VALUE REFERENCES

An expression at one entry can refer to the value of another entry, and the value of such an expression can be automatically recalculated when the value of the other entry changes. Value references are allowed in expressions wherever numbers are allowed. A value reference is made by either typing the coordinate of the desired entry (such as B5), or by "pointing" to the entry with the cursor (in this case, the coordinate will be "typed" automatically by VisiCalc). If an expression starts with a value reference, it must be preceded by a + character.

In order to insert the current value of another entry into an expression as a number, which will be unaffected by later changes to the other entry, type a value reference followed by the character # (e.g. B5#). If # is used by itself, it will be replaced by the current value of the expression stored in the entry you are changing.

FUNCTIONS

@SUM (*list*) Calculates the sum of the values in *list*. See **LISTS**, below.

@MIN (*list*) Calculates the minimum value in *list*.

@MAX (<i>list</i>)	Calculates the maximum value in <i>list</i> .
@COUNT (<i>list</i>)	Results in the number of non-blank entries in <i>list</i> . maximum number of entries in the list is 255.
@AVERAGE (<i>list</i>)	Calculates the average of the non-blank values in <i>list</i> . Maximum number of entries in the list is 255.
@NPV (<i>dr,range</i>)	Calculates the net present value of the cash flows in <i>range</i> , discounted at the rate specified by expression <i>dr</i> . The first entry in the range is the cash flow at the end of the first period, the second entry is the cash flow at the end of the second period, etc. See ENTRY RANGES , below.
@LOOKUP (<i>v,range</i>)	Compares the value <i>v</i> to the values of successive entries in <i>range</i> , and selects a corresponding value from the column or row immediately to the right or below the entries in <i>range</i> , as the result of the function. The values in <i>range</i> are normally in ascending order, and the result is the value corresponding to the last entry in <i>range</i> that is less than or equal to <i>v</i> before an entry greater than <i>v</i> is found. If the first entry in <i>range</i> is greater than <i>v</i> , the result of the function is NA.
@NA	Results in a "Not Available" value that makes all expressions using the value display as NA.
@ERROR	Results in an "Error" value that makes all expressions using the value display as ERROR
@PI	Results in 3.1415926536.
@ABS (<i>v</i>)	Results in the absolute value of <i>v</i> .
@INT (<i>v</i>)	Results in the integer portion of <i>v</i> .
@EXP (<i>v</i>) @SQRT (<i>v</i>)	Calculates the appropriate function.
@LN (<i>v</i>) @LOG10 (<i>v</i>)	The trigonometri calculations are done in radians.
@SIN (<i>v</i>) @ASIN (<i>v</i>)	
@COS (<i>v</i>) @ACOS (<i>v</i>)	
@TAN (<i>v</i>) @ATAN (<i>v</i>)	

EXAMPLES OF FUNCTIONS

@SUM (B4 . . . B15)
 @MIN (100, F4 . . . F11, @SUM (B4 . . . B15))
 @MAX (0, F4-F5)
 @NPV (.15, B4 . . . F4)

ENTRY RANGES

An entry range consists of a number of entries that are next to each other in a row or column, such as B2, B3, and B4, or B2, C2, D2, and E2. You enter an entry range by specifying the coordinate of the first entry in the range, then typing an ellipsis (. . . —you need only type the first period, VisiCalc will fill in the other), and then specifying the last entry. For example, the entry ranges just mentioned would be B2 . . . B4 and B2 . . . E2, respectively. Coordinates are specified by either typing them, or “pointing” to the desired entry with the cursor.

LISTS

A list consists of a series of expressions and ranges separated by commas. See the examples of lists in **EXAMPLES OF FUNCTIONS**, above.

COMMANDS

/B	Sets an entry to blank. Doesn't take effect unless you follow it with ←, →, or RETURN. Does not affect /F formats set at the entry.
/C	Clears the screen, sets all entries to blank, resets formats, windows, titles, etc. VisiCalc will wait for you to type a Y to confirm that you indeed want to erase all entries.
/D	Deletes the row (/DR) or column (/DC) on which the cursor lies.
/F	Sets the display format of an entry to one of the following formats: general (/FG), integer (/FI), dollars and cents (/F\$), left or right justified (/FL or /FR), or graph (/F*).
/FD	Resets an entry to use the global default format instead of an explicit format set with a /F command.
/GC	Sets the column width. Requests a number from 3 to 37; end with RETURN. The column width can be changed on a per window basis.
/GF	Sets the global default format that determines the display format of all entries without explicit format settings set with a /F command. Requests one of the same display formats used by the /F command. The global default format is a per window setting.
/GO	Sets the order of recalculation to be down the columns starting at entry A1 (/GOC), or across the rows starting at entry A1 (/GOR).

/GR	Sets recalculation to be automatic (/GRA) or manual (/GRM). You can always cause a manual recalculation of all entries by pressing the ! key.
/I	Inserts a row (/IR) or column (/IC) just above or to the left of the row or column on which the cursor lies.
/M	Moves an entire row or column to a new position. Prompts you to move the cursor from the row or column which you want to move to the destination row or column just before which the entries moved should reappear. End with RETURN.
/P	Print command. See PRINTING
/R	Replicate command. See REPLICATE
/S	Storage command. See STORAGE COMMANDS
/T	Sets a horizontal title area (/TH), a vertical title area (/TV), sets both a horizontal and vertical title area (/TB), or resets the window to have no title areas (/TN).
/W	Window control. Splits the screen into two windows at the cursor position (/WH for horizontal, /WV for vertical), or returns the screen to one window (/W1). Windows may be synchronized (/WS), or returned to unsynchronized (/WU).
/—	Repeating label. Requests the contents of a label entry; end with , , or RETURN. The contents of the label will be repeated over and over to fill the entry, no matter what the column width.

PRINTING

The /P command lets you output to the printer.

- 1) Position the cursor at the upper left corner of the rectangle of entries that you wish to print and type /P.
- 2) VisiCalc will prompt for the slot number of a printer or communications card; type the number followed by RETURN. If you press just RETURN, it will scan the slots and automatically pick one that has an appropriate device.
- 3) Lines printed by VisiCalc are usually terminated by the character pair RETURN/LINE FEED. To suppress the LINE FEED, type a minus (-) at this point. If you press + at this point, a RETURN will be output immediately.

- 4) If you want to output setup characters to the printer, type "", then type the characters, and end with RETURN.
- 5) Move the cursor to (or type the coordinates of) the lower right corner of the rectangle of entries to be printed out, and press RETURN.

Note: the screen may be overwritten by the printer controller. It will be returned to normal when the /P command is finished. You may stop printing at any time by pressing CTRL-C.

If, in step 2, you type D, VisiCalc will prompt you for a file name to receive the output as a text file. You respond in the same manner as for the /SS command (see **STORAGE COMMANDS**).

REPLICATE

The /R command allows you to make copies of entries.

- 1) Position the cursor on the first entry that you wish to replicate, and type /R.
- 2) VisiCalc will ask for the coordinates of the source (what you want to replicate). If you are just replicating the current entry, press RETURN. If you want to replicate a range of entries, type an ellipsis and provide a coordinate to complete an entry range specifying the source, ending with RETURN.
- 3) VisiCalc will ask for the coordinates of the target (where you want the copies to go). This may be a single coordinate or an entry range. End the target with RETURN. If you are replicating a source range of entries, the first source entry will be replicated into the target range, and succeeding source entries will be replicated into correspondingly succeeding target ranges.
- 4) If the expression being replicated contains value references VisiCalc will ask you, for each value reference, whether it should not be modified (respond by typing N), or should always refer to the entry in the same relative position (type R).

STORAGE COMMANDS

The /S command lets you save and load the current entries, using either diskette or cassette.

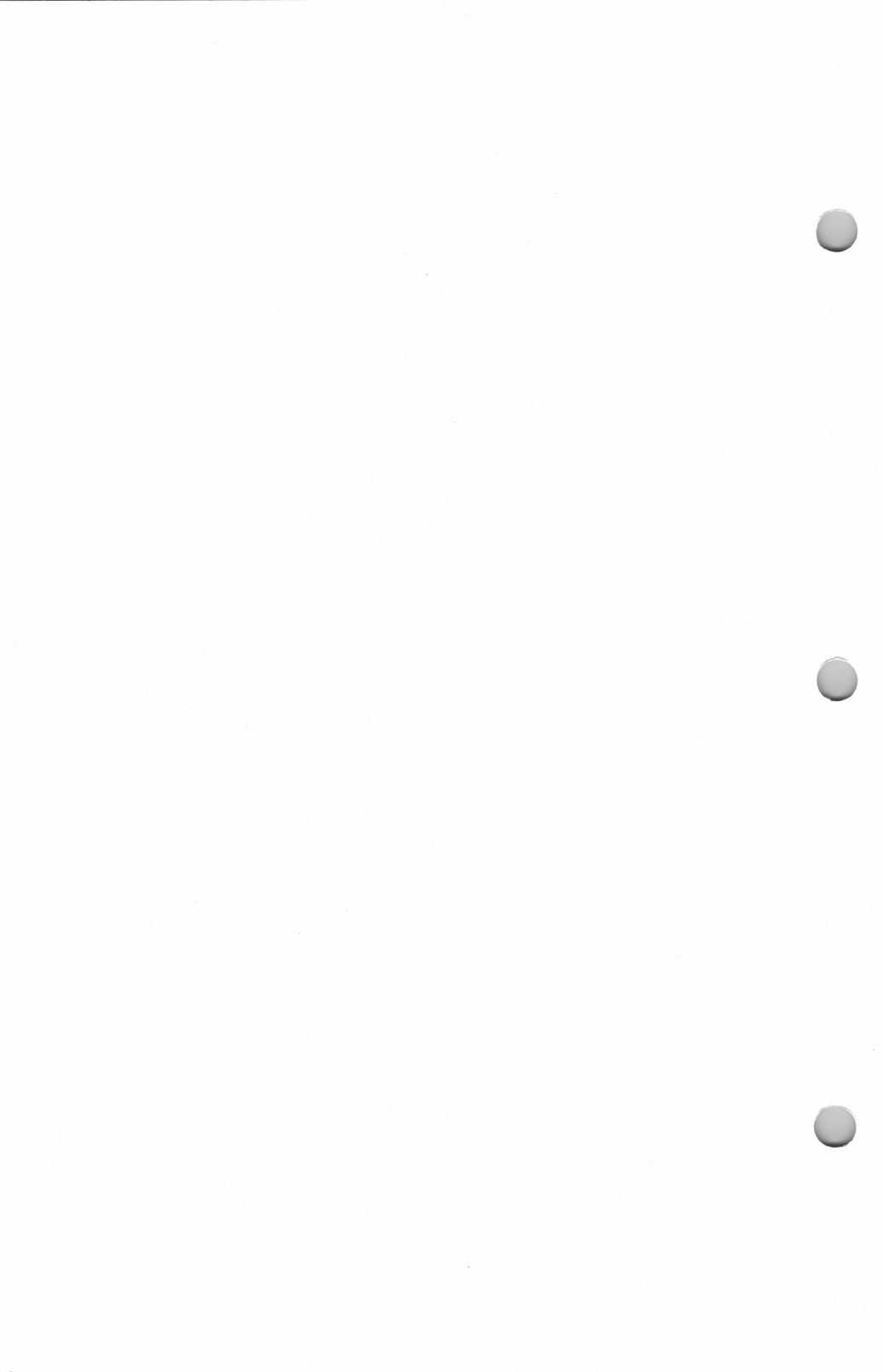
When using diskette, VisiCalc will prompt for a file name. You may either type a file name (with optional drive number and slot number, ended with RETURN), or just press RETURN (with a blank file name, and optional drive or slot specification such as ",D2"). If you don't provide a file name, VisiCalc will display the name of the first text file on the diskette. If that is the file name that you want to use, press RETURN; otherwise press and VisiCalc will show you the name of each successive text file on the diskette. When you have found the file name that you want, press RETURN. You may edit the file name before executing the com-

mand by typing additional characters to add to the name, and/or using the ESC key to erase part of it.

When using the cassette, VisiCalc will ask you to ready the recorder before it executes the command; press RETURN when ready.

The storage commands can be aborted by pressing CTRL-C.

/SS	Save all entries, titles, and window settings in a file on the diskette.
/SW	Save all entries, titles, and window settings on a cassette.
/SL	Load the contents of all entries that were saved in a file on the diskette. This command does no blank out all entries before doing the load; if that is desired, use the /C command first.
/SR	Load the contents of all entries that were saved on a cassette. This command does not blank out all entries before doing the load; if that is desired, use the /C command first.
/SD	Deletes the specified file on the diskette. Type Y to confirm.
/SI	Initializes, or formats, a blank diskette. Existing contents, if any, of the diskette will be erased. Displays default slot and drive numbers on the edit line. Edit, if desired, as described above for file names. Press RETURN to start initialization.



CPM

APPLE CP/M

DIGITAL RESEARCH

The following outline one CP/M is only reference material & a definite understanding of VOL.I & VOL.II of Microsoft Z80 Softcard Manual is required.

BOOTING APPLE CP/M

With Autostart ROM - With disk in Drive A:, turn power on.

Without Autostart ROM - Hit [RESET] Type 6 CRTL-K <CR>

Sign on Message:

Apple II CP/M

44 K Vers. 2.2x

(c) 1980 Microsoft

A>

For 56K System: You must have the Apple 16K Language Card or equivalent memory card in slot 0. First copy disk, then boot up the system as above and type:

CPM56 A:

The disk will be automatically updated, turn power off and reboot for a 56K system.

TO FORMAT A DISK

- All new disks must be formatted before use by running the program called format

A FORMAT d:

If d: = B: then formatting begins automatically

If d: = A: then you are asked to insert disk to be formatted in drive A.

TO COPY SYSTEM TRACKS 0-2 TO NEW DISKS

A formatted disk can be used for all disk drives except A:. Drive 'A': must contain the operating system on tracks 0-2. To copy these, put your system disk in Drive 'A': and type:

Copy d₁: = d₂: /S

d₁: is the drive you wish to put the CPM system on, and 'd₂:' is the drive you are copying the system from.

TO COPY FULL DISK

The full disk copy will overwrite your system tracks, so it is suggested that if you are coping a disk without a system track, that you put the the system on it after you run copy. To copy a full disk type:

Copy d₁: = d₂: d₁ = New Disk (slave)

 d₂ = Original Disk (master)

For a one drive system both d₁ and d₂ = A:

GENERAL OPERATIONAL NOTES FOR CP/M OPERATING SYSTEM

NEW KEYBOARD CHARACTERS

Type	New Character Available
CTRL-K	[(left bracket)
SHIFT-M	] (right bracket)
CTRL-@	RUBOUT
CTRL-B	\ (backslash)

KEYBOARD EDITING

←	Backspace and delete character
CTRL-X	Deletes entire line
CTRL-R	Retypes current line
CTRL-J	Same as return (with line feed)
CTRL-E	NON-RETURN line feed (Does not terminate entry)
RUBOUT (CTRL-@)	Deletes & echos last character typed

OUTPUT CONTROL

CTRL-S	Temporarily halts output. Any character hit except CTRL-S resumes output.
CTRL-P	Sends all output to printer as well as the CRT. Terminated with another CTRL-P.

CHANGING DISKS & CP/M WARM BOOT

Any CP/M disk can be read by merely changing disks, however if it must be written to then, it must be warm booted to read in directory information.

CTRL-C — Warm Boot on CP/M

FILE NAME SPECIFIERS

FILE NAME	Up to 8 characters with a 3 character extension
?	Stands for any single unspecified character.
*	Stands for any and all characters on that side of decimal point. Optional drive spec.

Examples: A:*.COM A:*.
 A:MB?????.*

afn = ambiguous file name ufn = unambiguous

BUILT IN CP/M COMMANDS

DIR Directory on Logged Drive

DIR d: Directory on Specified Drive

ERA afn - Erases Dir afn on logged drive

ERA *.* - Erases all user directory. Prompts all (Y/N)?

† Above file specifiers can also be used to Semi-Specify file name typed.

REN ufn = ufn 2 new name = old name
— renames file on logged drive

TYPE ufn FILE.NME

Lists on output device the contents of 'FILE.NME' which is on the currently logged disk

Save n ufn USER n

Saves n 256-byte blocks starting at memory location 100 HEX to disk under the file name 'FILE.EXT'.

USER n

n = 0 to 15 integer used to switch to a different user area of the disk

n:

Switch currently logged drive

PROGRAM DESCRIPTIONS

APDOS

Transfer program to move binary and text files from Apple DOS disks to CP/M disks

ASM

CP/M 8080 Assembler

CONFIGIO

Used to configure Apple CP/M to a particular system configuration

COPY

Used to copy entire disks or only system tracks 0-2. Will work with single drive systems.

DDT

Dynamic Debugging Tool. Allows modification, testing and debugging routines for 8080 assembly language programs

DOWNLOAD

Receiving program for transferring programs into Apple CP/M from external CP/M systems.

DUMP

Displays contents of a disk file in hexadecimal form.

ED

CP/M text editor

FORMAT

Formats a blank disk so that it can accept data. It does not put the system tracks on the disk.

LOAD

Used to convert files written by the assembler which have the extension '.HEX' into a '.COM' machine executable file.

MBASIC

Version 5.0 of Basic-80 which includes low resolution graphics.

PIP

File transfer program disk to disk or disk to I/O device

STAT

Status information

- Disk Capacity
- File Sizes
- File Indicators
- Device Assignments

SUBMIT

Allows CP/M commands and program input lines to be executed from a disk file instead of from the keyboard.

XSUB	Used with submit and allows character input from disk at all times during execution of programs.
CPM56	Configures 56K operating system
GBASIC	Same as MBASIC plus high resolution graphics
RW13	A two drive BIOS for transferring data from 13 sector CP/M disks to 16 sector CP/M disks.

COMMAND EDITING CONTROLS

rubout/delete

CTRL-@ delete and echo last character

CTRL-U or

CTRL-X delete line

CTRL-E continue on next line

CTRL-C reboot CP/M

CTRL-H backspace

CTRL-J

(line feed) terminate input

CTRL-M (car-

riage return) terminate command

CTRL-P printer on/off

CTRL-S stop/restart console output

EXTENSION TYPES

Extension	Type	Example
COM	Required Command file of a transient command (program)	PIP COM LOAD.COM
ASM	Required for assembly language source (text) files used with ASM command.	PROG1.ASM PATCH.ASM
PRN	Required for the listing file of the assembly language program.	PROG1.PRN PATCH.PRN
PRL	Required for MP/M relocatable programs.	RDT.PRL
HEX	Required for program file in "hex" format (machine language), which is ready to be LOAded.	PROG1.HEX PATCH.HEX
RSP	Required for MP/M "resident system programs."	SPOOL.RSP
BAS	Required for BASIC program source (text) files.	PROGBAS.BAS
INT	Required for BASIC program intermediate file for execution (already compiled).	PROGBAS.INT
BAK	Created by ED (text editor) as a backup copy of file before it is altered.	LETTER.BAK

\$\$\$	Temporary (scratch) files created and normally erased by ED and other programs	LETTER. \$\$\$
SUB	Text file with CP/M built-in or transient commands or programs; to be executed batch style by the SUBMIT program.	TRANSFORM SUB

TRANSIENT COMMANDS

STAT.COM Statistical information about file storage and device assignment.

ufn Unambiguous file name - specific file only

afn Ambiguous file name - can use file specifiers

STAT <cr> Gives remaining file space on disk in bytes and status
R/W = Read or Write to
R/O = Read Only
on currently logged drive

STAT d: afn.opt Same as above except for specified drive.

Returns the following:

RECS - Number of 128 byte records in the file BYTS

-File sizes in kilo bytes EX -Number of 16K extentions

D:FILENAME, TYP - File drive and file name

STAT afn.TYP \$S \$S causes size field to be displayed. (Virtual file size in records). For sequential files, both size and records fields are identical.

STAT DEV: STAT Ld 1 = pd 1, Ld 2 = pd 2, . . . , Ldn = pdn
Assigns physical device to logical device

STAT d:FILENAME.TYP\$R/O (AFN)

Marks file so that it is a read only file and cannot be written to or erased

STAT d:FILENAME.TYPE\$R/W (AFN)

Marks file for bothe reading and writing. Used to change R/O files to R/W

STAT d:FILENAME.TYP\$SYS

Attached system indicator to file and hides from display when DIR is called.

STAT d:FILENAME.TYP\$DIR

Removes system indicator and redisplayes the file when DIR is called.

STAT d:DSK Lists drive characteristics

d: Drive Characteristics	128: Records/Extent
1024: 128 Byte Record Capacity	8: Records/Block
128: Kilobyte Drive Capacity	32: Sectors/Track
48: 32 Byte Directory Entries	3: Reserved Tracks
48: Checked Directory Entries	

STAT USR: Produces a list of user numbers in use on that disk
Active User: Ø
Active Files: Ø

STAT VAL: Produces a summary of the available status commands

logical devices

CON: System Console
RDR: Paper Tape Reader
PUN: Paper Tape Punch
LST: Output List Device

physical devices

TTY: Teletype Device (slow speed console)
CRT: Cathode Ray Tube (high speed console)
BAT: Batch Processing (console is current RDR output goes to current LST:Device)
UC1: User Defined Console
PTR: Paper Tape Reader(high speed reader)
UR1: User Defined Reader #1
UR2: User Defined Reader #2
PTP: Paper Tape Punch (high speed punch)
UP1: User Defined Punch #1
UP2: User Defined Punch #2
LPT: Line Printer
UL1: User Defined List Device #1

STAT DEV: Produces list of current logical device on left and current physical device on right. i.e.:

CON: = CRT:
RDR: = URI:
PUN: = PTP:
LST: = TTY:

To change physical devices —

STAT 1d₁ = pd₁, 1d₂ = pd₂

Where: 1d = logical device pd = physical device

i.e.: CONSOLE to TTY

STAT CON: = TTY:

ED CP/M EDITOR

TEXT EDITOR FOR PREPARING '.ASM' FILES

ED ufn d:

D: option destination drive

CONTROL CHARACTERS

CTRL-C	Return to system (warm boot)
CTRL-E	Physical <cr> <lf> (screen only does not execute command)
CTRL-H	Backspaces cursor and erases last character typed.
CTRL-I	Logical TAB (Cols 1, 8, 15, . . .)
CTRL-J	Performs a RETURN. (linefeed)
CTRL-L	Logical <cr> <lf> in search and substitute strings.
CTRL-M	Performs a RETURN
CTRL-R	Retypes current line (types a clean line) without deleted characters
CTRL-U	Deletes current line. (types a '#' and goes to new line)
CTRL-X	Line delete (back space to beginning of line & delete characters)
CTRL-Z	String terminal
CTRL-P	Echo everything typed or displayed at the lineprinter
CTRL-S	Temporarily halts a long disp (strike any key to continue)
RUBOUT ←	Character delete
BREAK	Discontinue command

EDITOR COMMANDS

Command	Function
---------	----------

nA	Append n lines (or 1 line if no n) from the source file to the edit buffer. A "#" for n will append 65535 lines (fill the buffer), and a zero for n will append to fill half of the buffer (number of lines depends on size of your buffer and your system).
± nB	Move character pointer to beginning if +B, or end if -B of the buffer.
± nC	Move character pointer (+) n characters or backward (-) n characters. The CP counts a "carriage return" sequence as two characters (RETURN and LINE FEED).
± nD	Delete n characters forward (+), including the CP, or delete n characters backward (-), not including the CP. If no n, delete only the character pointed to by CP.
E	End ED session normally. The E command saves the buffered text and the rest of the source file text in a temporary output file, then renames the output file to the name of the source file (while copying the source file into a backup ".BAK" file to preserve the original source file). ED then terminates, bringing back the system.

nFstring $\left\{ \begin{array}{l} \uparrow Z \\ \swarrow \end{array} \right\}$ Find the string of characters n times (if no n is specified, find it once). F searches after the CP in the buffer and moves CP to the End of the found string. Follow string with Z terminator if you will add more ED commands; otherwise, use RETURN to terminate string.

H End ED session perform an E command, then execute ED again on the new source file (save your filek updates and return to edit again).

NOTE: you can substitute a "#" character for n in any of the ED commands; it gives n the highest value it can have -65535.

*CP is the character pointer.

I $\swarrow$ Insert a new line after the CP, moving CP to the end of last line inserted.

In CP/M 2.2:

— If upper case I is used in the command, all text inserted will be in upper case only.

— If lower case i is used, then the text will be inserted in upper and lower case.

ltext $\uparrow Z$ Insert characters after the CP, moving CP to the end of last character inserted.

nJ1 $\uparrow Z$ string2 $\uparrow Z$ string3

$\left\{ \begin{array}{l} \uparrow Z \\ \swarrow \end{array} \right\}$ Juxtapose strings of text by finding string1, inserting string2 at the end of string1, and delete all characters up to but not including string3 (juxtapose all three strings of text). CP is moved to the beginning of string3.

$\pm nK$ "Kill" (delete) the following (+) n lines, including the CP and characters following it on the current line, or delete the previous (—) n lines including the characters behind the CP on the current line.

$\pm nL$ Move CP to the beginning of the current line if n is zero; otherwise, move CP to the beginning of the current line and move it forward (+) n lines or backward (—) n lines.

nMstring $\left\{ \begin{array}{l} \uparrow Z \\ \swarrow \end{array} \right\}$ Repeat execution of the string of ED commands n times, if n is greater than 1. If n is zero or one, M will execute the string of commands repeatedly until an error occurs.

nNtext $\left\{ \begin{array}{l} \uparrow Z \\ \swarrow \end{array} \right\}$ Search for the nth occurrence of text throughout the buffer and source file (terminate text with RETURN or Z to add more commands). N moves the CP to the end of the found text. N will append source lines until it finds text.

O Omit ED session return to original source file.

$\pm nP$ Move CP, display and print pages of existing text in buffer. The n is the number of pages (24 lines per page) printed, where + n prints n pages following the CP, and — n prints n pages prior to the CP. OP (zero for n) will print the current line and page (first 23 lines following the current line). CP is moved to beginning of the printed page.

Q	Quit with no file alterations (leave the temporary file, source file and buffer file intact). Q returns you to the system. The ".BAK" backup file for the source file is not created, but if there was a previous ".BAK" file of the same name, it is deleted. (Watch out for this!)
R	Read from the file X\$\$\$\$\$.LIB and insert the lines following the CP, moving the CP to the end of the inserted lines (does not empty the ".LIB" file).
Rfilename	Read from the filename .LIB and insert the lines following the CP, moving the CP to the end of the inserted lines (does not empty the ".LIB" file).
nSoldtext ↑ Znewtext	
$\left\{ \begin{array}{c} \uparrow Z \\ \swarrow \end{array} \right\}$	Find oldtext in the buffer following CP and substitute new text for it; repeat the sequence n times if n is greater than one.
± nT	If n is not specified, or if n is 1, type (display the characters following the CP to the end of the line. If n is zero, type (display) the characters on the current line up to but not including the CP. If n is positive (+), display the following n lines including the current line. If n is negative (-), display the previous n lines not including the current line, and display the characters on the current line up to but not including the CP. The command sequence "B#T" will display the entire buffer.
± U	Translate all lower case characters input (typed in or inserted) to UPPER CASE if + U, or turn off translation by executing -U.
V	Turn on line number display (line numbers are not actually in the file) for lines in the buffer.
OV	Display the number of free bytes left in the buffer and the total memory size of the buffer (in decimal numbers). For example, in the display "27648/28832", "27648" is the number of bytes free, and "28832" are the number of bytes, total, in the current buffer.
nW	Write out to the temporary output file with the ".\$\$\$" extension the following n lines from the CP (including the current line). If no n, write out only the current line.
nX	Copies the following n lines of text to the file X\$\$\$\$\$.LIB (does not delete original lines). Retrieve lines using R command. If n is zero, this command will delete the file X\$\$\$\$\$.LIB.
**nZ	Suspend the ED program for n clock ticks (approximately n seconds).
± n	Perform a "± nLT" command sequence.
n:	absolute line number
n	Move CP to beginning of line number n.

N1 :n2 Specify a range of line numbers beginning with n1 and ending with n2. If either n1 or n2 is missing, substitute for it "the current line."

ERROR CONDITIONS

?	Unrecognized command
>	Memory buffer full
#	Command specified too many times
O	Cannot open LIB file in R command

ASM.COM 8080 Assembler - Creates a hex file from a previously written file in assembly code format.

To invoke type:

ASM FILENAME or ASM FILENAME.ASM

A file with the extension FILENAME.HEX is written.

ASM.COM CP/M ASSEMBLER

All specifications for the assembler must be understood when preparing your program with the editor. No command entries can be made externally, they must be included in the program writing.

PROGRAM FORMAT For Assembler Operation

LINE#	Label operation operand; comment
LINE#	Optional (not used by ASM)
LABEL	Identifier (alpha numerical, first character must be alpha) 16 character maximum. No spaces can be used, but \$ can be used to make more readable as ASM ignores it. A colon ':' can follow the label (ASM ignores it)
OPERATION	Assembler directive, pseudo operation, or 8080 machine operation code.
OPERAND	Formed out of constants and labels along with arithmetic and logical operations
COMMENT	Arbitrary comments after a semicolon ';'.
CONSTANT	16 bit value B(inary), O(ctal), Q(ctal), D(ecimal assumed), H(exadecimal)

ASSEMBLER DIRECTIVES

ORG exp	Set the program or data origin
END exp	End program, optional start address
EQU exp	Numeric "equates"
SET exp	Numeric "set"
IF exp	Begin conditional assembly
ENDIF	End of conditional assembly

DB exp	Define data bytes
DW exp	Define data words
DS exp	Define data storage areas

ERROR MESSAGES

D	Data error
E	Expression error
L	Label error
N	Not implemented
O	Overflow
P	Phase error - label does not have the same value on two subsequent passes through program
R	Register error - register value is not compatible with operator code
V	Value error - operand improperly framed
TERMINAL	No source file present
ERROR	No directory space
	Source file name error
	Source file read error
	Output file write error
	Cannot close file

LOAD.COM Converts a '.HEX' file into a '.COM' file

by producing a memory image file from the '.HEX' file. This is now machine executable code.
i.e.: LOAD d:FILENAME (.HEX REQUIRED)

DDT.COM CP/M DYNAMIC DEBUGGING TOOL

DDT filename.HEX filename.COM

EDITING

RUBOUT ←	Remove the last character typed
CTRL-X	Remove entire line, ready for retyping
CTRL-C	System reboot

COMMAND

s = Start Address	b = Bias Address
e = Ending Address	c = Fill Character
	d = Destination Add

A	Enter assembly language mnemonics with operands
---	---

A (start address)

D	Display memory in hexadecimal and terminated w/. ASCII
---	--

D<s>.<e>

F	fill memory with constant data
F<s>,<e>,<c>	
G	Begin execution with optional breakpoints.
G Gs Gs,b Gs,b,b G,b G,b,b	
I	Input a file (do not use spaces) .HEX on .COM
I FILENAME.TYP	
L	List memory using assembler mneumonics
L<s>,<e>	
M	Move memory segment from source to destination
M<s>,<e>,<d>	
R	Read program into DDT
R	
S	Substitute memory values. Terminated with a . (period).
S<s>	
T	Trace program execution
U	Untrace program monitoring
X	Examine and optionally alter the CPU state.
X<r>	

Where r is one of the 8080/Z80 CPU registers

C	Carry Flag	(0/1)
Z	Zero Flag	(0/1)
M	Minus Flag	(0/1)
E	Even Parity Flag	(0/1)
I	Interdigit Carry	(0/1)
A	Accumulator	(0-FF)
B	BC register pair	(0-FFFF)
D	DE register pair	(0-FFFF)
H	HL register pair	(0-FFFF)
S	Stack Pointer	(0-FFFF)
P	Program Counter	(0-FFFF)

PIP.COM CP/M PERIPHERAL INTERCHANGE PROGRAM

PIP only works within one user area at a time

Used to copy or transfer a file from one device to another

PIP <cr> Initiates PIP with command not specified. Prompt for entry is "*".

RESPONSE TO '*'

d1:FILE1.TYPE = d2:FILE2.TYP	Copy 'FILE2.TYP' to a new file called 'FILE1.TYP'
d1:FILE1.TYP = d2:FILE2.TYP, d3:FILE3.TYP	Concatenate files 'FILE2.TYP' and 'FILE3.TYP' and copy to 'FILE1.TYP'

FILE1.TYP = B:FILE2.TYP

A: = B:FILE2.TYP

A: = B:*.COM [V]

A: = B:*.*

PIP LST: = FILE.TYP

PIP PRN: = FILE.TYP

Move 'FILE2.TYP' from disk drive 'B:' to currently logged drive under the name 'FILE1.TYP'

Move 'FILE2.TYP' from disk drive 'B:' to disk drive 'A:' and use the same file name (FILE2.TYP)

Transfer all files with the extension '.COM' from disk drive 'B:' to disk drive 'A:' The optional [V], verify, checks to see if the copy is identical to the original.

Transfer all files from disk drive 'B:' to disk drive 'A:'

Sends 'FILE.TYP' to list device.

CAUTION: If tabs exist in file they are sent as tabs not as spaces, so if your printer does not have tabs specified it will tab incorrectly.

Same as LST: except that tabs are set at every eighth character position, lines are numbered and page ejects are inserted every 60 lines. It has an initial page eject.

KEYWORDS

NUL: send 40 "nulls" (ASCII code 0) to the device, usually a punch device for output. Example, where PROG.HEX is sent to the punch:

*PUN: = PROG.HEX, NULL:

EOF: send an end-of-file (ASCII Z) to the device (sent automatically by PIP during ASCII text file transfers, and only needed for special cases). Example:

*PUN: = NUL; X.ASM, EOF; NULL:

This example sends 40 nulls to the punch device, followed by a copy of the file X.ASM, followed by the end-of-file character (Z and 40 more nulls.

PRN: same as LST: (Send to the printer), except that tabs are expanded every eighth character, lines are numbered (as in the ED program), and page ejects (form feeds) are inserted every 60 lines (to advance the printer paper to the next page), with an initial page eject. Example:

*PRN: = SAMPLE.TXT

- INP:** special input device code which can be "patched" into the PIP program itself (you must write the patch in assembly language and add it to PIP). PIP receives the input character by character by calling a location in memory (103H) and storing the data starting at location 109H (parity bit must be zero—use the Z parameter).
- OUT:** special output device code which can be "patched" into the PIP program itself, like INP: described above. PIP calls location 106H and sends the data in register C (each character). Note to assembly language programmers: locations 109H through 1FFH of PIP memory image are not used and can be replaced with code for special-purpose device drivers (use DDT—the CP/M Debugger supplied by Digital Research with CP/M or MP/M). Examples:
- *GIZMO.CLK = INP;
(input from special device is stored in file GIZMO.CLK)
- OUT: = GIZMO.CLK
(copy of GIZMO.CLK is sent to the special device)

PIP PARAMETERS

- B** Block mode transfer. PIP puts data in a buffer until it reads an ASCII "x-off" character (S) from the device. PIP then clears the disk buffer and returns for more data. The size of the buffer depends on the size of your system (see the documentation provided with your system). Use this parameter to transfer data from a continuously reading device like a cassette player or reader. Example:
- *ENUFF.TXT = RDR: [B]
- Dn** PIP will delete characters which extend past column n (vertical columns on your terminal) while copying text files. Use this to truncate long lines if you are sending a file to a narrow device. Example:
- *PRN: = LONG.TXT[D52]
- E** Echo (redisplay) all copy operations on the terminal screen as they are being performed. Example:
- *COPY.TXT = SOURCE.TXT,S2.TXT,S3.TXT,S4.TXT[E]
- F** PIP will filter form feeds from the file (i.e., remove them) You can also use the P parameter to insert new form feeds.
- Gn** Get file from user area n (CP/M version 2.2 and MP/M).
- H** Hexadecimal data transfer: PIP checks all data for proper Intel hexadecimal format.
- I** Ignore ":00" records in the transfer of Intel hex format files (automatically sets the H parameter).
- L** Translate all upper-case characters to lower-case.

- N Add line numbers to each line copied into new file (starting at line 1). Each line number is followed by a colon. Leading zeros (e.g., 003) are deleted, unless you specify the "N2" parameter. "N2" leaves the leading zeroes and inserts a tab space after the numbers. You can expand tab spaces using the T parameter.
- O Object file transfer (for non-ASCII files): PIP ignores the physical end of the file during concatenation
- Pn PIP will include page ejects at every n line (with an initial page eject). If n is 1 (or if you don't specify n), page ejects occur every 60 lines. If you also use the F parameter, PIP removes the form feeds before inserting page ejects.
- Qstring ↑Z PIP will quit copying from the device or file when it finds the string of characters you specify (a string is a group of characters; e.g., STRING105%). You end your string with a Z (CTRL and Z simultaneously).
- R Read (copy) system (\$SYS) files; also performs a "W" parameter operation (CP/M version 2.2 and MP/M).
- Sstring ↑Z PIP will start copying from the device or file when it finds the string of characters you specify. End your string with a Z.
- Tn Expand the tab space to every nth column during the transfer of text files. you create a tab space in a text file using I; this parameter will expand the tab space from its usually fixed column amount.
- U Translate all lower-case characters to upper-case during the copying of the text files.
- V PIP will verify that data has been copied correctly by rereading the new copy file afterwards (copy file cannot be a device) and displaying a message if the copy was successful.
- W Overwrite (delete) read-only files (ignores the \$R/O attribute). (CP/M version 2.2 and MP/M only.)
- Z Turn the parity bit to zero on inputs of ASCII characters. Use this parameter especially if you are inputting from the INP: patch device.

Here are examples of PIP expressions with parameters:

*LST:=SAMPLE.TXT[NT8P60]

This expression sends the file SAMPLE.TXT to the list device (LST:), with line numbers, tabs expanded to every eighth character column, and page ejects at every 60 lines.

NOTE: the PRN: device assumes these parameters; if the listing device were assigned to PRN:, the above example could be rewritten:

*PRN:=SAMPLE.TXT

PIP.COM CP/M PERIFERIAL INTERCHANGE PROGRAM

DEVICE NAMES

Logical Devices

- CON: for "console" or terminal, including keyboard and display (Input/Output)
- RDR: for paper tape or card reader (input only)
- PUN: for paper tape or card punch (output only)
- LST: for "listing" device like a line printer (output only).

Physical Devices

- TTY: for a console or terminal, a reader, a punch, or a list device. (teletype)
- CRT: for a console or terminal, or list device (Cathode Ray Tube)
- PTR: for a paper tape or card reader device.
- PTP: for a paper tape or card punch device.
- LPT: for a list device (line printer).
- UC1: for a user-defined console or terminal.
- UR1: for a user-defined reader.
- UR2: for a second user-defined reader.
- UP1: for a user-defined output (punch) device.
- UP2: for a second user-defined output (punch) device.
- UL1: for a user-defined listing device.
- NOTE: BAT: is not included since it only re-assigns the values for RDR: and LST:

DUMP.COM Used to dump disk files to console

DUMP FILENAME.TYP (ufn)

Dumps the contents of the disk file to the console in hexadecimal form. The absolute byte address is listed to the left and followed by 16 bytes in hexadecimal form. The listing can be aborted by typing the Rubout Key (CTRL-@)

LOAD.COM Produces a memory image file which can be executed.

LOAD FILENAME.HEX

Loads file produced by ASM.COM and creates an executable file named 'FILENAME.COM'

SUBMIT.COM Used to batch CP/M commands for automatic processing

SUBMIT FILENAME

This program processes the commands found in the file called 'FILENAME.SUB' as if they were entered from the

keyboard. The file can contain variables by using \$1 \$2 \$3 ... \$n to represent defined variables

Examples: • A submit file called FILE.SUB is written by 'ED' to be listed as follows.

```
ASM $1
LOAD $2
REN $2 = $1.COM
```

- To start processing type submit file code copy CR
- The file called 'FILE.SUB' will now start, load the assembler file 'ASM.COM' and assemble the program 'CODE.ASM'. Then after assembly, load the program 'LOAD.COM' and process the 'CODE.HEX' file. Finally the 'CODE.COM' file will be renamed 'COPY'.

CP/M ERROR MESSAGES

D = Drive A, B, C, D, etc.

BDOS ERR on d: BAD SECTOR

1. Disk not in drive
2. Bad byte on disk
3. Disk drive controller malfunction
4. Recorded on drive aligned differently than one being used to read disk.

[CTRL-C] Returns to system

[RETURN] Ignores error and skips all data in that sector and continues with next sector. CAUTION: Your Data Will Be Incomplete.

BDOS ERR on d: SELECT

Drive selected does not exist

Hit any key to return to system.

BDOS ERR on d: READ ONLY

1. Writing to a disk that has been write protected.
2. New diskette inserted and CTRL-C not performed to change map to diskette.

To recover - hit any key and an automatic system boot will occur and change the diskette to read-write providing a mechanical override (disk protect) has not been used.

BDOS ERR on d: FILE R/O

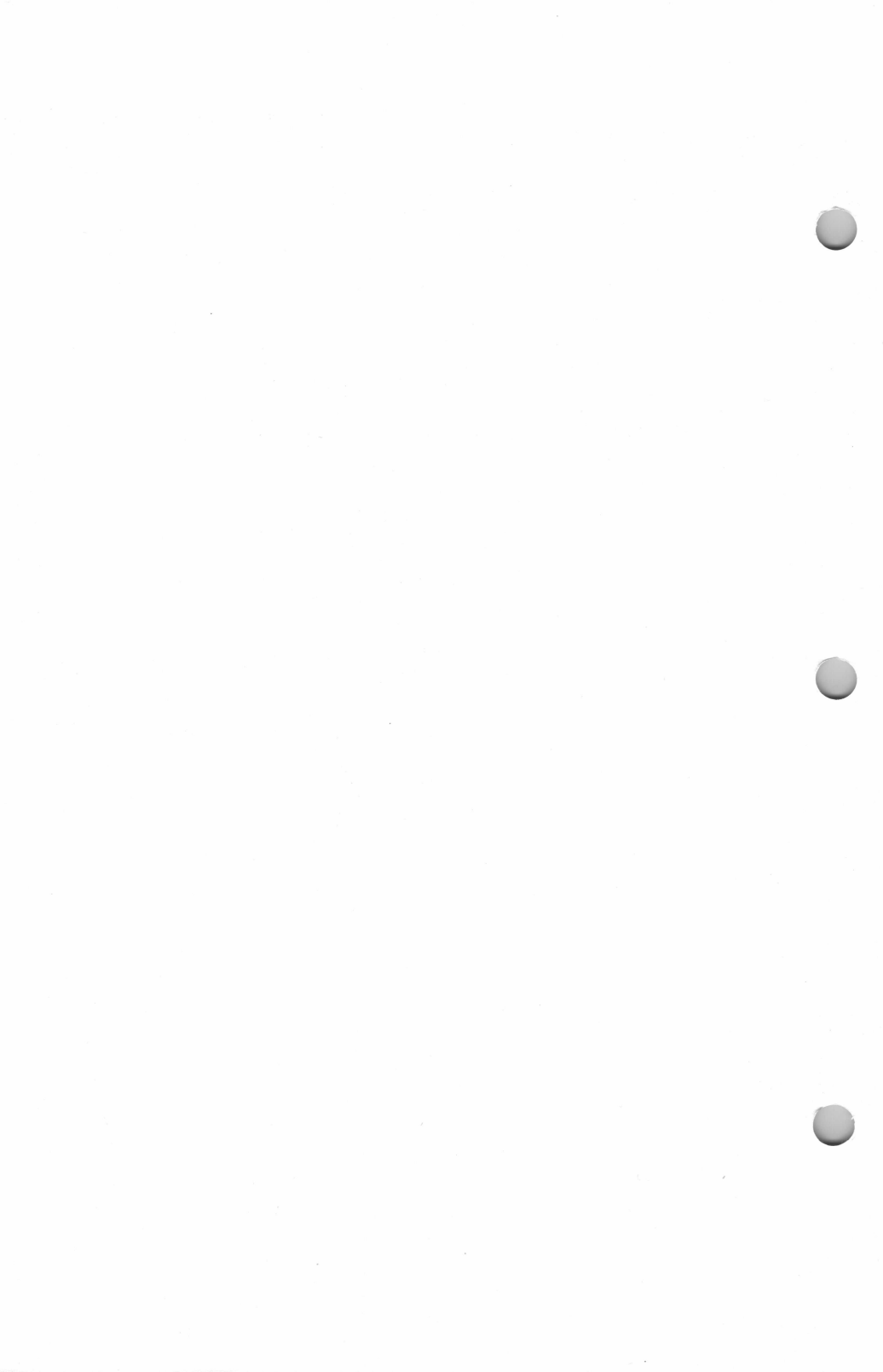
Ø Indicates that the file has been marked as read only by the stat command.

To recover - hit any key and the operation involving the read-only file is aborted.

FUNCTION SUMMARY

Function #	Description	Registers	
		Entry Parameter	Returned Value
1	Console Input	C: 01H	A: ASCII Character
2	Console Output	C: 02H	E: ASCII Character
3	Reader Input	C: 03H	A: ASCII Character
4	Punch Output	C: 04H	E: ASCII Character
5	List Output	C: 05H	E: ASCII Character
6	Direct Console I/O	C: 06H E: 0FFH (input) or char (output)	A: char or status (no value)
7	Get I/O Byte	C: 07H	A: I/O Byte Value
8	Set I/O Byte	C: 08H E: I/O Byte Value	
9	Print String	C: 09H DE: String Address	
10	Read Console Buffer	C: 0AH DE: Buffer Address	Console Character in Buffer
11	Get Console Status	C: 0BH	A: Console Status
12	Return Version Number	C: 0CH	HL: Version Number
13	Reset Disk System	C: 0DH	
14	Select Disk	C: 0EH E: Selected Disk	
15	Open File	C: 0FH DE: FCB Address	A: Directory Code
16	Close File	C: 10H DE: FCB Address	A: Directory Code
17	Search For First	C: 11H DE: FCB Address	A: Directory Code
18	Search For Next	C: 12H	A: Directory Code
19	Delete File	C: 13H DE: FCB Address	A: Directory Code
20	Read Sequential	C: 14H DE: FCB Address	A: Directory Code
21	Write Sequential	C: 15H DE: FCB Address	A: Directory Code
22	Make File	C: 16H DE: FCB Address	A: Directory Code
23	Rename File	C: 17H DE: FCB Address	A: Directory Code
24	Return Login Vector	C: 18H	HL: Login Vector

25	Return Current Disk	C: 19H	A: Current Disk
26	Set DMA Address	C: 1AH DE: DMA Address	
27	Get ADDR (ALLOC)	C: 1BH	HL: ALLOC Address
28	Write Protect Disk	C: 1CH	
29	Get Read/Only Vector	C: 1DH	HL: R/O Vector Value
30	Set File Attributes	C: 1EH DE: FCB Address	A: Directory Code
31	Get ADDR (Disk Parms)	C: 1FH	HL: DPB Address
32	Set/Get User Code	C: 20H E: OFFH (get or User Code (set))	A: Current Code or (no value)
33	Read Random	C: 21H DE: FCB Address	A: Return Code
34	Write Random	C: 22H DE: FCB Address	A: Return Code
35	Compute File Size	C: 23H DE: FCB Address	Random Record Field Set
36	Set Random Record	C: 24H DE: FCB Address	Random Record Field Set



BASIC-80

MICRO SOFT

BASIC—80 VERSION 5.0

CP/M BASIC

† - New Items in Basic-80 Over Microsoft Basic Rev. 4.51

🍏 - Apple MBasic-80 Only (Not in MBasic of Other Systems)

🍏 HG - GBasic Only

OPERATING PROCEDURE

To start BASIC, type under CP/M:

MBASIC [Return] <RET>

The system will reply: BASIC-80 Rev. 5.2
[Apple CP/M Version]
Copyright (c)1980 by Microsoft
Created 26 Aug. '80
14195 Bytes Free

Any BASIC command may now be entered

To start BASIC and specify a BASIC program to LOAD and RUN or to set options, type:

MBASIC[(filename)][(/F:n)][(/M:m)]

The /F option sets the maximum number of data files open during program execution. Default is 3.

The /M option sets the highest, memory address to be used if less than the maximum. Example:

MBASIC TEST/F:1/M:32767

SPECIAL CHARACTERS

Control @	Rubout
Control-A	Enters Edit Mode on the line being typed.
Control-B	Backslash
Control-C	Interrupts program execution and returns to BASIC-80 command level.
Control-G	Rings the bell at the terminal.
Control-H	Backspace. Deletes the last character typed.
Control-I	Tab. Tab stops are every eight columns.
Control-J	Line feed. Moves to next physical line.
Control-K	Right square bracket
Control-O	Halts program output while execution continues. A second Control-O restarts output.
Control-R	Retypes the line that is currently being typed.
Control-S	Suspends program execution.
Control-Q	Resumes program execution after a Control-S.
Control-X	Deletes the line that is currently being typed.

NOTE:	Control-@ Control B and Control K may be redefined using the CONFIGIO utility.
(return)	Ends every line typed in
(linefeed)	Used to break a logical line into physical lines
(rubout)	Erases last character typed
	Current line for EDIT,RENUM,DELETE,LIST,LLIST commands
&O or &	Octal constant
&H	Hexadecimal constant
:(colon)	Separates statements typed on the same line
?	Equivalent to PRINT
	(note: L? is not equivalent to LPRINT)

Variable Type Declaration Characters

		Storage Bytes Used
\$ String	(0 to 255 characters)	3 + # of characters
% Integer	(-32768 to 32767)	2
! Single precision	(7.1 digit floating point)	4
# Double precision	(16.8 digit floating point)	8

DEFINITIONS

File names are string expressions which follow the naming convention of the operating system.

For CP/M, filenames consist of an optional drive designator, A-F (the currently selected drive is used if not specified); a name of up to 8 characters; and an optional extension of up to 3 characters preceded by a ".".

Note: In the definitions of statements, functions and commands "string" means a string expression. All other parameters are numeric expressions ("exp") unless otherwise noted.

EXAMPLES

"A:TEXT,BAS"
 "DATA,INV"
 "RANDOM,FIL"

COMMANDS

Statement	Syntax/Function	Example
AUTO	AUTO [(LINE#) [, (INCREMENT)] Generates line number automatically after every carriage return	AUTO 100,10

BEEP



BEEP (PITCH), (DURATION) BEEP PDL(Ø),PDL(1)

Creates tone of specified
pitch and duration
PITCH - Ø is highest -255
is lowest DURATION - Ø is
shortest -255 is longest

CALL †
CALL%

CALL (VARIABLE)[(AUG-
MENT LIST)]

Variable & augment list
must contain only vars.

CALL &HFF3A
WRONG = 2 forms
CALL = 280
CALL% = 6502

CHAIN †

CHAIN [MERGE](FILENAME)
[,[(LINE NUMBER EXP)]
[,ALL][,DELETE(RANGE)]

To call a program & pass
variables to it from the
current program

CHAIN MERGE "OVERLAY
2", 1000 ,DELETE
1000-5000

CLEAR †

CLEAR (n)(,m)

Clear program variables
and set highest memory
location available to Basic
Optional second argu-
ment sets stack space for
Basic, Default 256 or 1/8
memory

CLEAR 100

CLEAR 32768, 2000

COLOR



COLOR = (COLOR
NUMBER)

0 black 8 brown
1 magenta 9 orange
2 dark blue 10 gray
3 purple 11 pink
4 dark green 12 green
5 gray 13 yellow
6 medium blue 14 aqua
7 light blue 15 white

COLOR = 13

COMMON †

COMMON (LIST OF
VARIABLES)

to pass variables to a
CHAINED program

COMMON A,B,C,D),G\$

CONT

CONT

Continue program execu-
tion

CONT

DEL* or
DELETE

DEL [(start line)][-(end
line))] Delete line(s) in a
program

DEL 20-25

*Changed Syntax from non-Apple systems using Basic-80

EDIT	EDIT n Edit a program line. See EDIT MODE subcom- mands.	EDIT 110
FILES	FILES (drive:; filename) List files in disk directory that match filename. ? matches any character. * matches any name or extension.	FILES FILES ""*BAS" FILES "TEST.BAS" FILES "B:*.?"
LIST	LIST [(start line) (-end line)] List program lines at terminal	LIST 100-1000
LLIST	LLIST [(start line) (-end line)] List program lines to printer	LLIST 50-250
LOAD	LOAD filename (,R) Load a program file. Extension of .BAS assumed. ,R option means RUN.	LOAD "INVEN"
MERGE	MERGE filename Merge program on disk with one in memory. Extension of .BAS assumed. File must have been SAV-Ed in ASCII mode.	MERGE "SUB1",
NAME	NAME old filename AS new filename Change the name of a disk file.	NAME "SUB1" AS "SUB2"
NEW	NEW Delete current program and variables.	NEW
RENUM	RENUM [first number (,first line(, increment))]	RENUM 100,,100
RESET	RESET Reinitialize CP/M disk information. Use when changing diskettes.	RESET
RUN	RUN (line number) Run a program (from line number) RUN filename (,R) Run a program, Extension of .BAS assumed. ,R used to keep files open	RUN RUN RUN "TEST"

SAVE	SAVE filename (.A) Save the program in memory with name "filename." .A saves program in ASCII. Extension of .BAS assumed.	SAVE "MYPROG"
SYSTEM	SYSTEM Return to CP/M (closes all files) SYSTEM may also be used as a program statement.	SYSTEM
*NOTRACE or TRON	NOTRACE Turn Trace off	NOTRACE
*TRACE or TROFF	TRACE Turn Trace on	TRACE
WIDTH	WIDTH n Set terminal width	WIDTH 80
*WIDTH [<LINE WIDTH>],[<SCREEN HEIGHT>]	WIDTH 25, 10 15-255 1-24 To set terminal line width in characters & screen height in lines	
*WIDTH LPRINT <LINE WIDTH>	15-255 Line width for printer	WIDTH LPRINT 80

EDIT MODE SUBCOMMANDS

Command	Function
A	Restart EDIT at the start of the line
nCc	Change n character(s)
nD	Delete n character(s) at the current position
E	End editing and save changes but don't type the rest of the line
H string (escape)	Delete the rest of the line and insert string
I string (escape)	Insert string at current position
nKc	Kill all characters up to the nth occurrence of c
L	Print the rest of the line and go to the start of the line
Q	Quit and cancel all changes
nSc	Search for nth occurrence of c
X string (escape)	Got o ght end of the line and insert string
(rubout) or (backspace)	Delete previous character on the line
(return)	End editing and save changes

*Changed Syntax from non-Apple systems using Basic-80

PROGRAM STATEMENTS (except I/O)

Statement	Syntax/Function	Example
DEF	DEFFNx(argument list) = expression Define an arithmetic or string function	DEF FNA(X,Y) = SQR (X*X* + Y*Y)
	DEF USRn = address Define the entry address for the nth assembly language subroutine.	DEF USR3 = &2000

Note: The following forms of DEF declare that all variable names starting with one of the letters given as an argument will be given the specified type as a default. letter 1-letter2 means all letters between letter1 and letter2.

DEFDBL	DEFDBL letter(-letter) . . . [,letter)-letter]) Define double precision default variable names	DEFDBL A, B, E-G
DEFINT	DEFINT letter(-letter) . . . Define integer default variable names	DEFINT I-N
DEFSNG	DEFSNG letter(-letter) . . . Define single precision default variable names	DEFSNG X-Z
DEF USR	DEF USR [(DIGIT)] = (integer expression) Specify the starting address of an assembly language routing in decimal.	DEF USR0 = 24000
DEFSTR	DEFSTR letter(-letter) . . . Define string default variable names	DEFSTR S-V
DIM	DIM variable [size1,size2 . . .)] . . . Allocate space for arrays. Arrays may be dimensioned dynamically.	DIM A (3), B\$(10,2,3) DIM Z (2*1)
END	END Stop program and return to BASIC command level.	END
ERASE	ERASE variable (,variable) . . . Release space previously reserved for an array	ERASE A,B\$

ERROR	ERROR code Generate error of code (see tabl. May call user ON ERROR routine or force BASIC to handle er- ror.	ERROR 17
FOR	FOR variable = expression TO expression (STEP expression) Used with NEXT statement to repeat a sequence of program lines. The variable is in- cremented by the value of STEP.	FOR I = 1 to 5 STEP .5 . . .
GOSUB	GOSUB line number Call a BASIC subroutine by branching to the specified line number. See RETURN.	GOSUB 210
GOTO	GOTO Branch to specified line number	GOTO 90
GR	GR (screen#),(color#) Initialize low-resolution graphics mode. Screen Screen Number Mode Ø 40x40 graphics + 4 lines text 1 40x48 graphics with no lines text	GR 1,15
HLIN	HLIN(x1 coordinate),x2 coordinate)at(y coor- dinate) In low resolution graphics to draw a horizontal line from point (x1,y) to point (xz,y) x1 = Ø to 39 column x2 = Ø to 39 column y = Ø to 47 row	
HOME	HOME Clear the screen of all text and move the cursor to the upper left corner	HOME
HTAB	HTAB (screen position number)	HTAB 13

IF	IF expression THEN statement (:statement) . . (ELSE statement . . .) The relation X > Y is tested. If true, the THEN clause is executed. If false, the ELSE clause is executed.	IF X > Y THEN Y = X ELSE Y = A
LET	(LET) variable = expression Assign a value to a variable	LET X = 1 + 5
NEXT	NEXT variable(,variable).. Delimits the end of a FOR loop	NEXT I
ON ERROR GOTO	ON ERROR GOTO line number Enables error trap subroutine beginning at specified line. If line number = 0, disables error trapping. If line number = 0 inside error trap routine, forces BASIC to handle error.	ON ERROR GOTO 1000
ON...GOSUB	On expression GOSUB line(,line). . . GOSUBs to statement specified by expression. (If J + 1 = 1, to 20; if J + 1 = 2, to 20; if J + 1 = 3, to 40)	ON J + 1 GOSUB 20,20,40
ON...GOTO	ON expression GOTO line(,line).. Branches to statement specified by I (To 20 if I = 1, to 30 if I = 2)	ON I GOTO 20,30
IF...GOTO	If <expression> GOTO <line number> If expres- sion = 0 then GOTO line number	IF x GOTO 100
*OUT (Not available on Apple)	OUT port, byte Puts byte 16 + 1 to output port 41	OUT 41 16 + 1
POKE	POKE location, byte Puts byte specified into memory location specified	POKE &23100, 255

*Changed Syntax from non-Apple systems using Basic-80

POP	POP Return from a subroutine that was branched to by a GOSUB without branching back to the statement following the most recent GOSUB	POP
RANDOMIZE †	RANDOMIZE [(expression)] To reseed the random number generator	RANDOMIZE
REM	REM any text Allows user to insert comments in program (not executed). Note: "." does not terminate a REM statement.	REM comment
RESTORE	RESTORE (line number) Resets DATA pointer so that DATA statements may be re-read.	RESTORE
RESUME	RESUME or RESUME O Returns from ON...ERROR routine to statement that caused the error	RESUME
	RESUME NEXT Returns to statement after the one that caused the error	RESUME NEXT
	RESUME line Returns to the specified line	RESUME 100
RETURN	RETURN Return from subroutine to statement, following last GOSUB performed	RETURN
STOP	STOP Stop program execution, print BREAK message, and return to COMMAND mode	STOP
SWAP	SWAP variable, variable Exchanges values of two variables (The variables must be of the same type.)	SWAP A\$,B\$

*WAIT	WAIT addr, mask, (select)	WAIT 21,1
(non-Apple)	WAIT address , I,[J]	WAIT GHE000, 128
	OR'ed with J & AND'ed with I	

WHILE...-
WEND

While (expression)

•
•
•
•

[(loop statements)]

WEND

Execute a series of
statements in a loop as
long as a guide condition
is true.

PRINT USING Format Field Specifiers

Numeric Possible Field

Specifier	Digits	Characters	Definition	Example
#	1	1	Numeric field	####
.	0	1	Decimal point	#,.
+	0	1	Print leading or trailing sign. Positive numbers will have "+", negative numbers will have "-".	###+
-	0	1	Trailing sign - prints "-" if negative, otherwise blank.	##,##-
**	2	2	Leading asterisk fill	*###,##
\$\$	1	2	Floating dollar sign. \$ is placed in front of the leading digit.	\$\$###,##
**\$	2	3	Asterisk fill and floating dollar sign	**\$,##
,	1	1	Use comma every three digits (left of decimal point only)	##,###,##
++++	0	4	Exponential format. Number is aligned so leading digit is non-zero.	#,####

String

!	Single character	!
(spaces)	2+ numbers of spaces	
	character field	

*Changed Syntax from non-Apple systems using Basic-80

INPUT OUTPUT STATEMENTS

Statement	Syntax/Function	Example
CLOSE	CLOSE [(#)f,(#)f...] Closes disk files. If no argument, all open files are closed.	CLOSE 6
DATA	DATA item (,item...) Specifies data to be used in a READ statement	DATA2,3 "PLUS",4
FIELD	FIELD (#) f, count AS string variable (,count AS string variable)... Define feilds in a random file buffer	FIELD#1, 3 AS A\$
GET#	GET (#) f (;record number) Read a random file record	GET#1 17*1
GET	Read keyboard character	GETI\$
INPUT	INPUT("prompt string literal";) variable (,variable)... Read data from the terminal	INPUT "VALUES", A,B
INPUT#	INPUT#f,variable (,variable)... Read data from a disk file	INPUT #2,A,B
INVERSE	INVERSE Set video output mode for black characters on white background	INVERSE
KILL	KILL filename Delete a disk file	KILL "INVEN.BAS"
LINE INPUT	LINE INPUT ("prompt string literal",) string-variable Read an entire line from the terminal	LINE INPUT \$A
LINE INPUT#	LINE INPUT#f, string-variable Read an entire line from a disk file	LINE INPUT#2,B\$
LSET	LSET field variable = expression Store data in random file buffer, left justified	LSET A\$ = "JOHN JONES"
NORMAL	NORMAL Restore video output to usual white charactes on black background	NORMAL

OPEN	OPEN mode-string, (#) file number, filename Open a disk file. Mode string must be one of: I = input file O = output file R = random file	OPEN "O", #1, "OUTPUT"
OPTION BASE †	OPTION BASE Where N is 1 or 0 To declare the minimum value for array subscripts	OPTION BASE 1
PDL 🍏	PDL(I) I = 0 to 3 integer Returns current value in the range of 0 to 255 of the game control specified by I	A FUNCTION
PLOT 🍏	PLOT (x coordinate), (y coordinate) x coordinate = 0 to 39 y coordinate = 0 to 47 In low resolution graphics mode to place a dot at (x coordinate) and (y coordinate)	PLOT 24,39
PRINT & PRINT USING	PRINT (USING format string;) exp(exp)... Print data at the terminal using the format specified. See table for format characters.	PRINT USING "!" A\$,B\$
PRINT# & PRINT#1 USING	PRINT#f, (USING format string;)exp(exp)... Write data to a disk file	PRINT #4,A,B
LPRINT & LPRINT USING	LPRINT (USING format;) variable(,variable)... Write data to a line printer	LPRINT A,B
PUT	PUT (#)f(,record number) Writes data from a random buffer to a data file	PUT#3,4
READ	READ variable(,variable)... Reads data into specified variables from a DATA statement	READ I,X,A\$

RSET	RSET field variable = expression Puts data into a random file buffer, right justified	RSET B\$ = "CORRECT"
TEXT 	TEXT Reset the screen to normal full Apple text mode (24 line x 40 characters)	TEXT
VLIN 	VLIN (y1 coordinate), (y2 coordinate) at (x coordinate) y1 = 0 to 47 row y2 = 0 to 47 row X = 0 — 39 column In low resolution graphics mode to draw a vertical line from point (x,y1) to point (x,y2)	VLIN20,45 at 12
VTAB 	VTAB (screen line number) Move cursor to the line specified	VTAB 12
WRITE †	WRITE [(list of expressions)] To output data at the terminal lists numerics separated by a comma and strings using quote marks.	WRITE A,B,C\$
WRITE# †	WRITE# (file number), (list of expression) to write data to a sequential file writes commas & quotation marks to disk as stated above.	
VPOS 	VPOS(x) Returns the current vertical position of the cursor. The topmost position is 1. x is a dummy argument	

G BASIC ADDITIONS

Available only in G Basic

HGR  HG	HGR (screen number), (color number) Screen number = integer 0 to 3 Color number = integer 0 to 12	HGR 1,2
---	---	---------

Initialize high - resolution
graphics mode

HCOLOR



HG

HCOLOR = (color
number)

HCOLOR = 7

0 black 7 white
1 green 8 black1
2 violet 9 white1
3 white 10 black 2
4 black 11 white 2
5 orange 12 reverse
6 blue

Set the color for plotting
in high resolution graphics

HPlot



HG

HPlot
[(x1),(y1)][to(x2),(y2)-
...[to(xn),(yn)]] Plot a
point or draw line(s) on
the high resolution
screen, using points
specified by
(x1,y1),(x2,y2), etc.

HSCRN



HG

HSCRN (x,y)
In high resolution graphics
mode, Returns -1 if a dot
exists at point (x,y)

PRINT HSCRN(46,100)

OPERATORS

Symbol

Function

=

Assignment, or equality test

-

Negation or subtraction

+

Addition or string concatenation

*

Multiplication

/

Division (floating point result)

†

Exponentiation

Integer division (integer result)

MOD

Integer modulus (integer result)

NOT

One's complement (integer)

AND

Bitwise AND (integer)

OR

Bitwise OR (integer)

XOR

Bitwise exclusive OR (integer)

EQV

Bitwise equivalence (integer)

IMP

Bitwise implication (integer)

=,(<,>,<=,>=)

Relational tests (result is TRUE = -1
or FALSE = 0)

(=,<,>,<=,>=)

(=,<,>,<=,>=)

()

The precedence of operators is:

- (1) Expressions in parentheses
- (2) Exponentiation ($A \square B$)
- (3) Negation ($-X$)
- (4) $*, /$
- (5)
- (6) MOD
- (7) $+, -$
- (8) Relational operators [$=, (,), (,), (=,)=$]
- (9) NOT
- (10) AND
- (11) OR
- (12) XOR
- (13) IMP
- (14) EQV

ARITHMETIC FUNCTIONS

Statement	Function	Example
ABS (exp)	Absolute value of expression	$Y = \text{ABS} (A + B)$
ATN (exp)	Arctangent of the expression (in radians)	PRINT ATN (A)
BUTTON 	I is in the range of 0 to 3 Returns the current value of the push button on the game controller, specified by I. 0 = button not depressed 1 = button depressed	PDL HSCRN(X,Y)
CDBL (exp)	Convert the expression to a double precision number	$A = \text{CDBL} (Y)$
CINT (exp)	Convert the expression to an integer	$B = \text{CINT} (R)$
COS (exp)	Cosine of the expression (expression in radians)	$A = \text{COS} (2.3)$
CSNG (exp)	Convert the expression to a single precision number	$C = \text{CSNG} (X)$
ERL	Error line number	PRINT ERL
ERR	Error code number	PRINT ERR
EXP (exp)	Raises the constant e to the power of the variable	$B = \text{EXP} (C)$

FIX (exp)	Returns truncated integer of expression	J = FIX (A/B)
FRE (exp)	Gives memory free space not used by BASIC	PRINT FRE(O)
*INP (port) (not available on Apple)	Inputs a byte from an Input port	PRINT INP (21)
INT (exp)	Evaluates the expression for the largest integer contained	C = INT (X + 3)
LOG (exp)	Gives the natural logarithm of the expression	D = LOG (Y-2)
LPOS (dummy)	Carriage position of line	LPOS (3)
PEEK (add)	Reads a byte from memory at address specified	PRINT PEEK (62000)
POS (dummy)	Carriage position of terminal	POS(5)
RND (parameter)	Generates a random number. Parameter (0 seed new sequence =0 return previous random number)0 return new random number	E = RND (1)
SCRN  HG	SCRN (x,y) x = 0 to 39 column Y = 0 to 47 row Returns the color of the point at (x,y)	Print SCRN(10,15)
SGN (exp)	1 if expression > 0 0 if expression = 0 -1 if expression < 0	B = SGN (X + Y)
SIN (exp)	Sine of the expression (expression in radians)	B = SIN (A)
SPC (exp)	Used in PRINT statement to print spaces	PRINT SPC (5)
SQR (exp)	Square root of the expression	C = SQR (D)
TAB (exp)	Used only in PRINT statement. Tabs carriage to specified position.	PRINT TAB (20)

TAN (exp)	Tangent of the expression (expression in radians)	D = TAN (3.14)
USRn (exp)	Calls the user's machine language subroutine with the specified argument. See DEFUSR.	PRINT USR (G27000)
VARPTR (var)	Returns address of variable in memory or zero if variable has not been assigned a value	I = VARPTR(X)
VARPTR (#file number exp)	Returns the address of the 128 byte data buffer for file number specified	J = VARPTR (#2)

STRING FUNCTIONS

Statement	Function	Example
ASC(string)	Returns the ASCII value of the first character of a string	PRINT ASC (A\$)
CHR\$(exp)	Returns a one-character string whose charater has the ASCII code of exp	PRINT CHR\$ (48)
FRE(string)	Returns remaining string free space	PRINT FRE (A\$)
HEX\$(exp)	Converts a number to a hexadecimal string	H\$ = HEX\$(100)
INKEY \$	INKEY \$ Returns either a one character string containing a character read from the terminal or a null string if no character is pending at the terminal. No characters will be echoed and all characters are passed through to the program except for Control-C which ter- minates the program.	
INPUT\$ (length [, (#file number)]	Returns a string of length characters read from con- sole Characters are not echoed	X\$ = INPUT\$(4)
	Returns a string of length characters from a disk file.	X\$ = INPUT\$(5, #2)

INSTR[(exp, string 1, string 2)	Returns the first position of the first occurrence of string 2 in string 1 starting at position exp	INSTR(A\$,":") INSTR(3,X\$,Y\$)
LEFT\$(string, length)	Returns leftmost length characters of the string expression	B\$ = LEFT\$(X\$,8)
LEN(string)	Returns the length of a string	PRINT LEN(B\$)
MID\$(string, start [,length])	Returns characters from the middle of the string starting at the posi- tion specified to the end of the string or for length characters MID\$ may also be used to assign a substring in- side a string	A\$ = MID\$(X\$,5,1]) MID\$(A\$,3,2) = "TO"
OCT\$(exp)	Converts a number to an octal string	O\$ = OCT\$(100)
RIGHT\$(string, length)	Returns rightmost length characters of the string expression	C\$ = RIGHT\$(X\$,8)
SPACE\$(exp)	Returns a string of exp spaces	S\$ = SPACE\$(20)
STR\$(exp)	Converts a numeric expression to a string	PRINT STR\$(35)
STRING\$(count,string)	Returns a string count long containing first character of string	X\$ = STRING\$(100,"A")
STRING\$(count,exp)	Returns a string count long containing characters with numeric value exp	Y\$ = STRING\$(100,42)
VAL(string)	Converts the string representation of a number to its numeric value	PRINT VAL("3,1")

INPUT OUTPUT FUNCTIONS

Name	Function	Syntax
CVD	Converts an 8-character string to a double precision number	CVD (8-character string)
CVI	Converts a 2-character string to an integer	CVI (2-character string)

CVS	Converts a 4-character string to a single precision number	CVS (4-character string)
EOF	Returns true (-1) if a file is positioned at its end	EOF (file number)
LOC	Next record number to read (random file). Number of sectors read or written (sequential file.)	LOC (file number)
LOF	Returns the number of records present in the last extent read or written. If the file does not exceed one extent (128 records), the LOF returns the true length of the file.	LOF (file number)
MKD\$	Converts a double precision number to an 8-character string	MKD\$ (double precision value)
MKI\$	Converts an integer to a 2-character string	MKI\$(integer value)
MKS\$	Converts a single precision number to a 4-character string	MKS\$ (single precision value)

TABLE OF ERROR CODES*

ERR = ERROR NUMBER

ERR = LINE IN PROGRAM ON WHICH ERROR OCCURS

Code (Err) Error

1	NEXT without FOR
2	Syntax error
3	RETURN without GOSUB
4	Out of data
5	Illegal function call
6	Overflow
7	Out of memory
8	Undefined line
9	Subscript out of range
10	Redimensioned array
11	Division by zero
12	Illegal direct
13	Type mismatch
14	Out of string space
15	String too long
16	String formula too complex
17	Can't continue
18	Undefined user function
19	No RESUME
20	RESUME without error

21	Unprintable error
22	Missing operand
23	Line buffer overflow
26	FOR without NEXT
29	WHILE without WEND
30	WEND without WHILE
31	Reset Error
32	Graphics statement not implemented

Disk Errors

50	Field overflow
51	Internal error
52	Bad file number
53	File not found
54	Bad file mode
55	File already open
57	Disk I/O error
58	File already exists
61	Disk full
62	Input past end
63	Bad record number
64	Bad file name
66	Direct statement in file
67	Too many files
68	Disk read only
69	Drive select error
70	File read only

*NOTE: See also ERL, ERR, ON ERROR GOTO RESUME and ERROR

*NULL	Not impemented in Apple version but in other systems. Sends specified nulls after every return.	NULL 10
-------	--	---------

STANDARD KEYBOARD CHARACTER ADDITIONS

(May be changed with CONFIGIO)

CTRL-K	— [
CTRL-@	— RUB (rubout)
CTRL-U	— CTRL I (tab)
→	— CTRL I (tab)
CTRL-B	— \

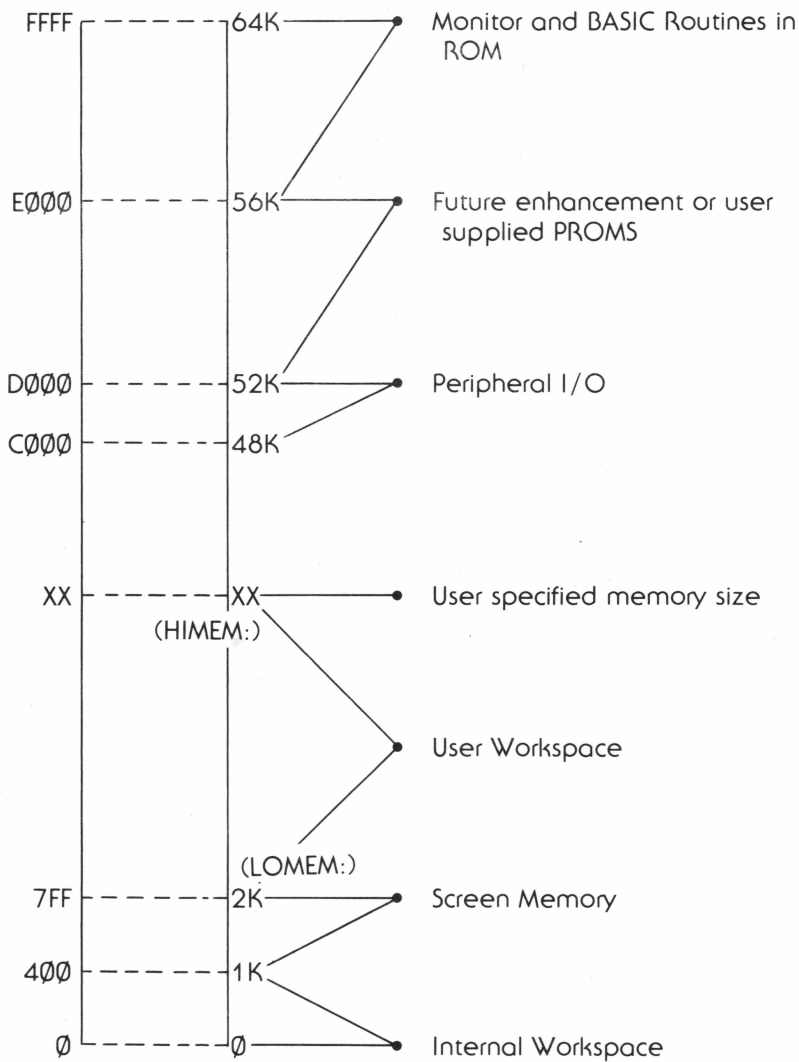
*Changed Syntax from non-Apple systems using Basic-80

HARDWARE

Memory Map

Memory Range	Description
0.1FF	Program work space; not available to user.
200.2FF	Keyboard character buffer.
300.3FF	Available to user for short machine language programs.
400.7FF	Screen display area for page 1 text or color graphics.
800.2FFF	In cassette tape version, the APPLESOFT BASIC interpreter
800.XXX	If firmware APPLESOFT (Part number A2B00009X) installed, user program and variable space, where XXX is maximum RAM memory to be used by APPLESOFT. This is either total system RAM memory, or less if the user is reserving part of high memory for machine language routines or high-resolution screen buffers.
2000.3FFF	Firmware APPLESOFT only: high-resolution graphics display page 1.
3000.XXX	Cassette tape APPLESOFT II; user program and variables where XXX is maximum available RAM memory to be used by APPLESOFT. This is either total system RAM memory, or less if the user is reserving part of high memory for machine language routines or page 2 high-resolution graphics.
4000.5FFF	High-resolution graphics display page 2.
C000.CFFF	Hardware I/O Addresses.
D000.DFFF	Future ROM expansion.
D000.F7FF	APPLESOFT II firmware version, with select switch "ON" (up).
E000.F7FF	APPLE Integer BASIC.
F800.FFFF	APPLE System Monitor.

Simplified Memory Map



APPLESOFT VARIABLE MAPS

SIMPLE VARIABLES

POINTERS

\$69-\$6A

REAL		INTEGER		STRING POINTERS	
NAME (pos)	1st byte (pos) 2nd byte	NAME (neg)	1st byte (neg) 2nd byte	NAME (neg)	1st byte (pos) 2nd byte
exponent	1 byte		high byte	length	1 byte
mantissa	m.s. byte		low byte	address	low byte
mantissa		0		address	high byte
mantissa		0			0
mantissa	l.s. byte	0			0

ARRAY VARIABLES

\$6B-\$6C

REAL		INTEGER		STRING POINTERS	
NAME (pos)	1st byte (pos) 2nd byte	NAME (neg)	1st byte (neg) 2nd byte	NAME (neg)	1st byte (pos) 2nd byte
OFFSET pointer to next variable: add to address of this variable name low byte high byte		OFFSET pointer to next variable: add to address of this variable name low byte high byte		OFFSET pointer to next variable: add to address of this variable name low byte high byte	
NO. OF DIMENSIONS one byte		NO. OF DIMENSIONS one byte		NO. OF DIMENSIONS one byte	
SIZE Nth DIMENSION high byte low byte		SIZE Nth DIMENSION high byte low byte		SIZE Nth DIMENSION high byte low byte	
SIZE 1st DIMENSION high byte low byte		SIZE 1st DIMENSION high byte low byte		SIZE 1st DIMENSION high byte low byte	
REAL (0, 0, . . . , 0) exponent 1 byte mantissa m.s. byte mantissa mantissa mantissa l.s. byte		INTEGER% (0, 0, . . . , 0) high byte low byte		STRING\$ (0, 0, . . . , 0) length 1 byte address low byte address high byte	
REAL (N, N, . . . , N) exponent 1 byte mantissa m.s. byte mantissa mantissa mantissa l.s. byte		INTEGER* (N, N, . . . , N) high byte low byte		STRING\$ (N, N, . . . , N) length 1 byte address low byte address high byte	

\$6D-\$6E

Strings are stored in order of entry, from HIMEM: down. String table points to first character of each string, at the bottom of the string in memory. As strings are changed, new pointing addresses are written; when available memory is used up, house-cleaning deletes all abandoned strings. (House-cleaning is forced by a FRE(X)).

All arrays are stored with the right-most index ascending slowest; e.g., the numbers in the array A*(1, 1) where A*(0, 0)=0, A*(1, 0)=1, A*(0, 1)=2, A*(1, 1)=3 would be found in memory in proper sequence.

Applesoft Zero Page Usage

Location(s) (in hex)	Use
\$0-\$5	Jump instructions to continue in APPLESOFT. (reset 0G return for APPLESOFT is equivalent to reset ctrl C return for Integer BASIC.)
\$A-\$C	Location for USR function's jump instruction.
\$D-\$17	General purpose counters/flags for APPLESOFT.
\$20-\$4F	APPLE II system monitor reserved locations.
\$50-\$61	General purpose pointers for APPLESOFT.
\$62-\$66	Result of last multiply/divide.
\$67-\$68	Pointer to beginning of program. Normally set to \$0801 for ROM version, or \$3001 for RAM (cassette tape) version.
\$69-\$6A	Pointer to start of simple variable space. Also points to the end of the program plus 1 or 2, unless changed with the LOMEM: statement.
\$6B-\$6A	Pointer to beginning of array space.
\$6D-\$6E	Pointer to end of numeric storage in use.
\$6F-\$70	Pointer to start of string storage. Strings are stored from here to the end of memory.
\$71-\$72	General pointer.
\$73-\$74	Highest location in memory available to APPLESOFT plus one. Upon initial entry to APPLESOFT, is set to the highest RAM memory location available.
\$75-\$76	Current line number of line being executed.
\$77-\$78	"Old line number". Set up by a ctrl C, STOP or END statement. Gives line number at which execution was interrupted.
\$79-\$7A	"Old text pointer". Points to location in memory for statement to be executed next.
\$7B-\$7C	Current line number from which DATA is being READ.
\$7D-\$7E	Points to absolute location in memory from which DATA is being READ.
\$7F-\$80	Pointer to current source of INPUT. Set to \$201 during an INPUT statement. During a READ statement is set to the DATA in the program it is READING from.
\$81-\$82	Holds the last-used variable's name.
\$83-\$84	Pointer to the last-used variable's value.
\$85-\$9C	General usage.
\$9D-\$A3	Main floating point accumulator.

\$A4	General use in floating point math routines.
\$A5-\$AB	Secondary floating point accumulator.
\$AC-\$AE	General usage flags/pointers.
\$AF-B0	Pointer to end of program (not changed by LOMEM:)
\$B1-\$C8	CHRGET routine. APPLESOFT calls here everytime it wants another character.
\$B8-\$B9	Pointer to last character obtained through the CHRGET routine.
\$C9-\$CD	Random number.
\$D0-\$D5	High-resolution graphics scratch pointers.
\$D8-\$DF	ONERR pointers/scratch.
\$E0-\$E2	High-resolution graphics X and Y coordinates.
\$E4	High-resolution graphics color byte.
\$E5-\$E7	General use for high-resolution graphics.
\$E8-\$E9	Pointer to beginning of shape table.
\$EA	Collision counter for high-resolution graphics.
\$F0-\$F3	General use flags.
\$F4-\$F8	ONERR pointers.

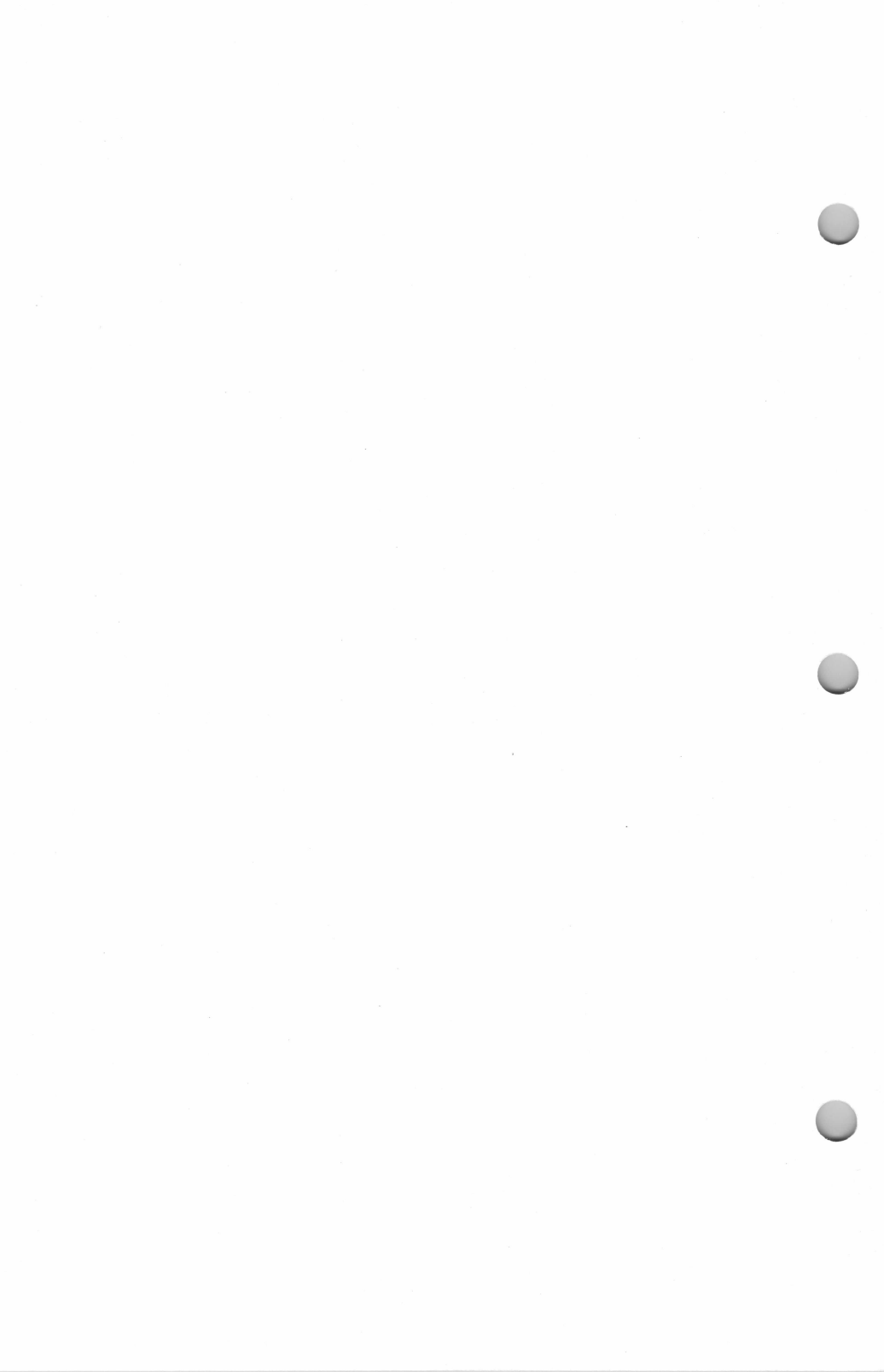


Table 1: Keyboard Special Locations				
Location:				
Hex	Decimal		Description	
\$C000	49152	-16384	Keyboard Data	
\$C010	49168	-16368	Clear Keyboard Strobe	

Table 2: Keys and Their Associated ASCII Codes									
Key	Alone	CTRL	SHIFT	Both	Key	Alone	CTRL	SHIFT	Both
space	\$A0	\$A0	\$A0	\$A0	RETURN	\$8D	\$8D	\$8D	\$8D
0	\$B0	\$B0	\$B0	\$B0	G	\$C7	\$87	\$C7	\$87
1!	\$B1	\$B1	\$A1	\$A1	H	\$C8	\$88	\$C8	\$88
2"	\$B2	\$B2	\$A2	\$A2	I	\$C9	\$89	\$C9	\$89
3#	\$B3	\$B3	\$A3	\$A3	J	\$CA	\$8A	\$CA	\$8A
4\$	\$B4	\$B4	\$A4	\$A4	K	\$CB	\$8B	\$CB	\$8B
5%	\$B5	\$B5	\$A5	\$A5	L	\$CC	\$8C	\$CC	\$8C
6&	\$B6	\$B6	\$A6	\$A6	M	\$CD	\$8D	\$DD	\$9D
7'	\$B7	\$B7	\$A7	\$A7	N	\$CE	\$8E	\$DE	\$9E
8(	\$B8	\$B8	\$A8	\$A8	O	\$CF	\$8F	\$CF	\$8F
9)	\$B9	\$B9	\$A9	\$A9	P@	\$D0	\$90	\$C0	\$80
:*	\$BA	\$BA	\$AA	\$AA	Q	\$D1	\$91	\$D1	\$91
;+	\$BB	\$BB	\$AB	\$AB	R	\$D2	\$92	\$D2	\$92
,<	\$AC	\$AC	\$BC	\$BC	S	\$D3	\$93	\$D3	\$93
=	\$AD	\$AD	\$BD	\$BD	T	\$D4	\$94	\$D4	\$94
.>	\$AE	\$AE	\$BE	\$BE	U	\$D5	\$95	\$D5	\$95
/?	\$AF	\$AF	\$BF	\$BF	V	\$D6	\$96	\$D6	\$96
A	\$C1	\$81	\$C1	\$81	W	\$D7	\$97	\$D7	\$97
B	\$C2	\$82	\$C2	\$82	X	\$D8	\$98	\$D8	\$98
C	\$C3	\$83	\$C3	\$83	Y	\$D9	\$99	\$D9	\$99
D	\$C4	\$84	\$C4	\$84	Z	\$DA	\$9A	\$DA	\$9A
E	\$C5	\$85	\$C5	\$85	→	\$88	\$88	\$88	\$88
F	\$C6	\$86	\$C6	\$86	←	\$95	\$95	\$95	\$95
					ESC	\$9B	\$9B	\$9B	\$9B

All codes are given in hexadecimal. To find the decimal equivalents, use Table 3.

Table 3: The ASCII Character Set									
Decimal:		128	144	160	176	192	208	224	240
Hex:		\$80	\$90	\$A0	\$B0	\$C0	\$D0	\$E0	\$F0
0	\$0	nul	dle		0	@	P		p
1	\$1	soh	dc1	!	1	A	Q	a	q
2	\$2	stx	dc2	"	2	B	R	b	r
3	\$3	etx	dc3	#	3	C	S	c	s
4	\$4	eot	dc4	\$	4	D	T	d	t
5	\$5	enq	nak	%	5	E	U	e	u
6	\$6	ack	syn	&	6	F	V	f	v
7	\$7	bel	etb	'	7	G	W	g	w
8	\$8	bs	can	(	8	H	X	h	x
9	\$9	ht	em	)	9	I	Y	i	y
10	\$A	lf	sub	*	:	J	Z	j	z
11	\$B	vt	esc	+	;	K	[	k	{
12	\$C	ff	fs	,	<	L	\	l	
13	\$D	cr	gs	-	=	M	]	m	}
14	\$E	so	rs	.	>	N	^	n	~
15	\$F	si	us	/	?	O	_	o	rub

Table 4: Video Display Memory Ranges

Screen	Page	Begins at:		Ends at:	
		Hex	Decimal	Hex	Decimal
Text/Lo-Res	Primary	\$400	1024	\$7FF	2047
	Secondary	\$800	2048	\$BFF	3071
Hi-Res	Primary	\$2000	8192	\$3FFF	16383
	Secondary	\$4000	16384	\$5FFF	24575

Table 5: Screen Soft Switches

Location:			Description:
Hex	Decimal		
\$C050	49232	-16304	Display a GRAPHICS mode.
\$C051	49233	-16303	Display TEXT mode.
\$C052	49234	-16302	Display all TEXT or GRAPHICS.
\$C053	49235	-16301	Mix TEXT and a GRAPHICS mode.*
\$C054	49236	-16300	Display the Primary page (Page 1).
\$C055	49237	-16299	Display the Secondary page (Page 2).
\$C056	49238	-16298	Display LO-RES GRAPHICS mode.*
\$C057	49239	-16297	Display HI-RES GRAPHICS mode.*

Table 6: Screen Mode Combinations

Primary Page			Secondary Page		
Screen	Switches		Screen	Switches	
All Text	\$C054	\$C051	All Text	\$C055	\$C051
All Lo-Res	\$C054	\$C056	All Lo-Res	\$C055	\$C056
Graphics	\$C052	\$C050	Graphics	\$C052	\$C050
All Hi-Res	\$C054	\$C057	All Hi-Res	\$C055	\$C057
Graphics	\$C052	\$C050	Graphics	\$C052	\$C050
Mixed Text and Lo-Res	\$C054	\$C056	Mixed Text and Lo-Res	\$C055	\$C056
	\$C053	\$C050		\$C053	\$C050
Mixed Text and Hi-Res	\$C054	\$C057	Mixed Text and Hi-Res	\$C055	\$C057
	\$C053	\$C050		\$C053	\$C050

Table 7: ASCII Screen Characters

Table 7-1: 160K Screen Characters																		
Inverse					Flashing				Normal								(Lowercase)	
Decimal	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240		
Hex	\$00	\$10	\$20	\$30	\$40	\$50	\$60	\$70	\$80	\$90	\$A0	\$B0	\$C0	\$D0	\$E0	\$F0		
0 50	@	P		0	@	P		0	@	P		0	@	P		0		
1 51	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	!	1		
2 52	B	R	"	2	B	R	"	2	B	R	"	2	B	R	"	2		
3 53	C	S	#	3	C	S	#	3	C	S	#	3	C	S	#	3		
4 54	D	T	\$	4	D	T	\$	4	D	T	\$	4	D	T	\$	4		
5 55	E	U	%	5	E	U	%	5	E	U	%	5	E	U	%	5		
6 56	F	V	&	6	F	V	&	6	F	V	&	6	F	V	&	6		
7 57	G	W	'	7	G	W	'	7	G	W	'	7	G	W	'	7		
8 58	H	X	(	8	H	X	(	8	H	X	(	8	H	X	(	8		
9 59	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	)	9		
10 5A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	*	:		
11 5B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	+	;		
12 5C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	,	<		
13 5D	M	]	-	=	M	]	-	=	M	]	-	=	M	]	-	=		
14 5E	N	^	.	>	N	^	.	>	N	^	.	>	N	^	.	>		
15 5F	O	_	/	?	O	_	/	?	O	_	/	?	O	_	/	?		

Table 7. ASCII Screen Character Set

Table 8: Low-Resolution Graphics Colors

Decimal	Hex	Color	Decimal	Hex	Color
0	\$0	Black	8	\$8	Brown
1	\$1	Magenta	9	\$9	Orange
2	\$2	Dark Blue	10	\$A	Grey 2
3	\$3	Purple	11	\$B	Pink
4	\$4	Dark Green	12	\$C	Light Green
5	\$5	Grey 1	13	\$D	Yellow
6	\$6	Medium Blue	14	\$E	Aquamarine
7	\$7	Light Blue	15	\$F	White

Table 9: Annunciator Special Locations

Ann.	State	Address:	
		Decimal	Hex
0	off	49240	-16296 SC058
	on	49241	-16295 SC059
1	off	49242	-16294 SC05A
	on	49243	-16293 SC05B
2	off	49244	-16292 SC05C
	on	49245	-16291 SC05D
3	off	49246	-16290 SC05E
	on	49247	-16289 SC05F

Table 10: Input/Output Special Locations

Function:	Address:		Read/Write
	Decimal	Hex	
Speaker	49200	-16336 SC030	R
Cassette Out	49184	-16352 SC020	R
Cassette In	49256	-16288 SC060	R
Annunciators*	49240 through 49247	-16296 through -16289	SC058 through SC05F
Flag inputs	49249	-16287 SC061	R
	49250	-16286 SC062	R
	49251	-16285 SC063	R
Analog Inputs	49252	-16284 SC064	R
	49253	-16283 SC065	
	49254	-16282 SC066	
	49255	-16281 SC067	
Analog Clear	49264	-16272 SC070	R/W
Utility Strobe	49216	-16320 SC040	R

Table 11: Text Window Special Locations

Function:	Location:		Minimum/Normal/Maximum Value	
	Decimal	Hex	Decimal	Hex
Left Edge	32	\$20	0/0/39	\$0/\$0/\$17
Width	33	\$21	0/40/40	\$0/\$28/\$28
Top Edge	34	\$22	0/0/24	\$0/\$0/\$18
Bottom Edge	35	\$23	0/24/24	\$0/\$18/\$18

Table 12: Normal/Inverse Control Values		
Value:		Effect:
Decimal	Hex	
255	\$FF	COUT will display characters in Normal mode.
63	\$3F	COUT will display characters in Inverse mode.
127	\$7F	COUT will display letters in Flashing mode, all other characters in Inverse mode.

Table 13: Autostart ROM Special Locations		
Location:		Contents:
Decimal	Hex	
1010	\$3F2	Soft Entry Vector. These two locations contain the address of the reentry point for whatever language is in use. Normally contains \$E003.
1011	\$3F3	
1012	\$3F4	Power-Up Byte. Normally contains \$45. See below.
64367 (-1169)	\$FB6F	This is the beginning of a machine language subroutine which sets up the power-up location.

Table 14: Page Three Monitor Locations				
Address:		Use:		
Decimal	Hex	Monitor ROM	Autostart ROM	
1008	\$3F0	None.	Holds the address of the subroutine which handles machine language "BRK" requests (normally \$FA59).	
1009	\$3F1			
1010	\$3F2	None.	Soft Entry Vector.	
1011	\$3F3			
1012	\$3F4	None.	Power-up Byte.	
1013	\$3F5	Holds a "JuMP" instruction to the subroutine which handles Applesoft II "&" commands.* Normally \$4C \$58 \$FF.		
1014	\$3F6			
1015	\$3F7			
1016	\$3F8	Holds a "JuMP" instruction to the subroutine which handles "USER" (<u>CTRL Y</u>) commands.		
1017	\$3F9			
1018	\$3FA			
1019	\$3FB	Holds a "JuMP" instruction to the subroutine which handles Non-Maskable Interrupts.		
1020	\$3FC			
1021	\$3FD			
1022	\$3FE	Holds the address of the subroutine which handles Interrupt ReQuests.		
1023	\$3FF			

Table 15: Mini-Assembler Address Formats

Mode:	Format:
Accumulator	None.
Immediate	$\#S\{value\}$
Absolute	$S\{address\}$
Zero Page	$S\{address\}$
Indexed Zero Page	$S\{address\}, X$ $S\{address\}, Y$
Indexed Absolute	$S\{address\}, X$ $S\{address\}, Y$
Implied	None.
Relative	$S\{address\}$
Indexed Indirect	$(S\{address\}, X)$
Indirect Indexed	$(S\{address\}), Y$
Absolute Indirect	$(S\{address\})$

Table 16: RAM Organization and Usage

Page Number:		Used For:		
Decimal	Hex			
0	\$00	System Programs		
1	\$01	System Stack		
2	\$02	GETLN Input Buffer		
3	\$03	Monitor Vector Locations		
4	\$04	Text and Lo-Res Graphics Primary Page Storage		
5	\$05			
6	\$06			
7	\$07			
8	\$08	Text and Lo-Res Graphics Secondary Page Storage	FREE	
9	\$09			
10	\$0A			
11	\$0B			
12 through 31	\$0C \$1F			RAM
32 through 63	\$20 \$3F	Hi-Res Graphics Primary Page Storage		
64 through 95	\$40 \$5F	Hi-Res Graphics Secondary Page Storage		
96 through 191	\$60 \$BF			

Table 17: ROM Organization and Usage			
Page Number:		Used By:	
Decimal	Hex		
208	\$D0	Programmer's Aid #1	Applesoft II BASIC
212	\$D4		
216	\$D8		
220	\$DC	Integer BASIC	
224	\$E0		
228	\$E4		
232	\$E8		
236	\$EC		
240	\$F0		
244	\$F4	Utility Subroutines	
248	\$F8	Monitor ROM	Autostart ROM
252	\$FC		

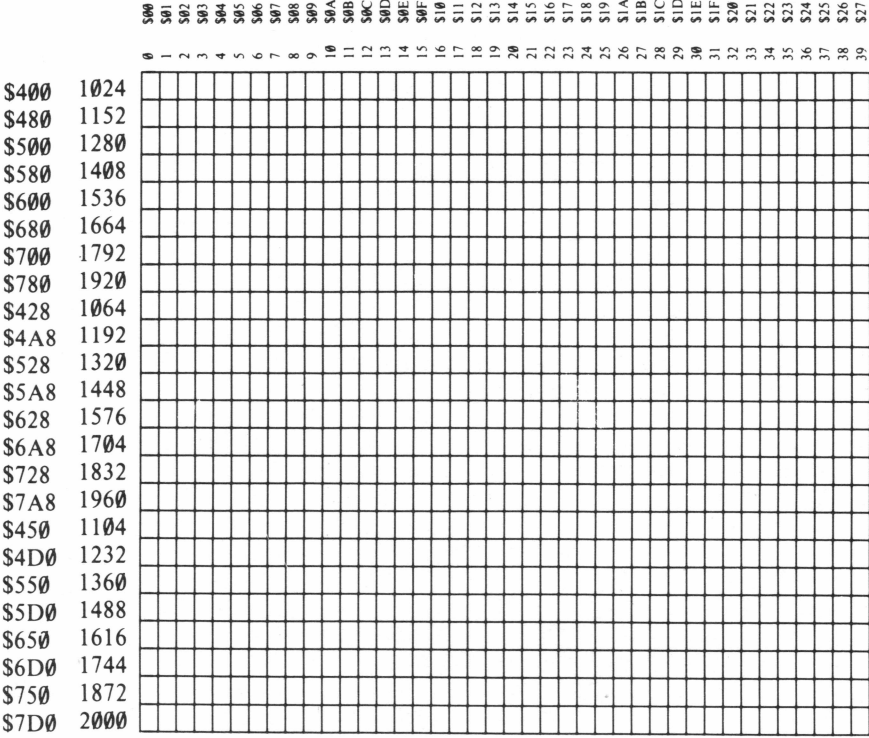


Figure 1. Map of the Text Screen

		\$00
		\$01
		\$02
		\$03
		\$04
		\$05
		\$06
		\$07
		\$08
		\$09
		\$0A
		\$0B
		\$0C
		\$0D
		\$0E
		\$0F
		\$10
		\$11
		\$12
		\$13
		\$14
		\$15
		\$16
		\$17
		\$18
		\$19
		\$1A
		\$1B
		\$1C
		\$1D
		\$1E
		\$1F
		\$20
		\$21
		\$22
		\$23
		\$24
		\$25
		\$26
		\$27
		\$28
		\$29
		\$2A
		\$2B
		\$2C
		\$2D
		\$2E
		\$2F
		\$30
		\$31
		\$32
		\$33
		\$34
		\$35
		\$36
		\$37
		\$38
		\$39
		\$3A
		\$3B
		\$3C
		\$3D
		\$3E
		\$3F
		\$40
		\$41
		\$42
		\$43
		\$44
		\$45
		\$46
		\$47
		\$48
		\$49
		\$4A
		\$4B
		\$4C
		\$4D
		\$4E
		\$4F
		\$50
		\$51
		\$52
		\$53
		\$54
		\$55
		\$56
		\$57
		\$58
		\$59
		\$5A
		\$5B
		\$5C
		\$5D
		\$5E
		\$5F
		\$60
		\$61
		\$62
		\$63
		\$64
		\$65
		\$66
		\$67
		\$68
		\$69
		\$6A
		\$6B
		\$6C
		\$6D
		\$6E
		\$6F
		\$70
		\$71
		\$72
		\$73
		\$74
		\$75
		\$76
		\$77
		\$78
		\$79
		\$7A
		\$7B
		\$7C
		\$7D
		\$7E
		\$7F
		\$80
		\$81
		\$82
		\$83
		\$84
		\$85
		\$86
		\$87
		\$88
		\$89
		\$8A
		\$8B
		\$8C
		\$8D
		\$8E
		\$8F
		\$90
		\$91
		\$92
		\$93
		\$94
		\$95
		\$96
		\$97
		\$98
		\$99
		\$9A
		\$9B
		\$9C
		\$9D
		\$9E
		\$9F
		\$A0
		\$A1
		\$A2
		\$A3
		\$A4
		\$A5
		\$A6
		\$A7
		\$A8
		\$A9
		\$AA
		\$AB
		\$AC
		\$AD
		\$AE
		\$AF
		\$B0
		\$B1
		\$B2
		\$B3
		\$B4
		\$B5
		\$B6
		\$B7
		\$B8
		\$B9
		\$BA
		\$BB
		\$BC
		\$BD
		\$BE
		\$BF
		\$C0
		\$C1
		\$C2
		\$C3
		\$C4
		\$C5
		\$C6
		\$C7
		\$C8
		\$C9
		\$CA
		\$CB
		\$CC
		\$CD
		\$CE
		\$CF
		\$D0
		\$D1
		\$D2
		\$D3
		\$D4
		\$D5
		\$D6
		\$D7
		\$D8
		\$D9
		\$DA
		\$DB
		\$DC
		\$DD
		\$DE
		\$DF
		\$E0
		\$E1
		\$E2
		\$E3
		\$E4
		\$E5
		\$E6
		\$E7
		\$E8
		\$E9
		\$EA
		\$EB
		\$EC
		\$ED
		\$EE
		\$EF
		\$F0
		\$F1
		\$F2
		\$F3
		\$F4
		\$F5
		\$F6
		\$F7
		\$F8
		\$F9
		\$FA
		\$FB
		\$FC
		\$FD
		\$FE
		\$FF

Figure 3. Map of the High-Resolution Graphics Screen

In each box:

	0	\$0000
	1024	\$0400
	2048	\$0800
	3072	\$0C00
	4096	\$1000
	5120	\$1400
	6144	\$1800
	7168	\$1C00

To obtain the address for any byte, add the addresses for the byte's box row, box column, and position in box.

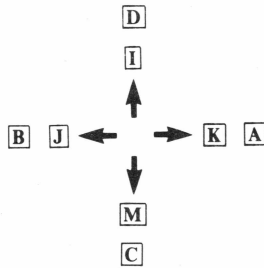
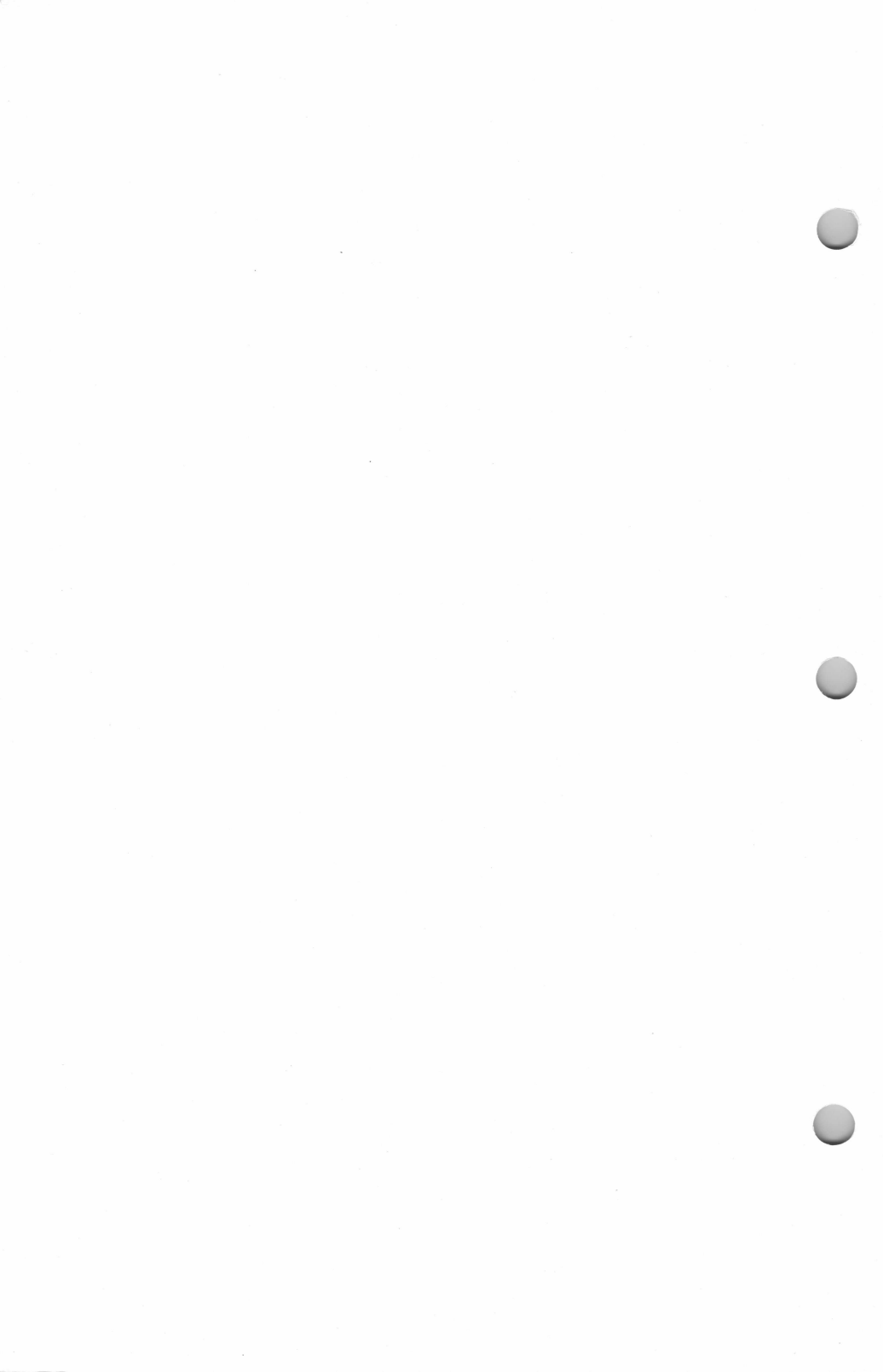


Figure 4. Cursor-moving Escape Codes.

System Memory Map		
Page Number:		
Decimal	Hex	
0	\$00	RAM (48K)
1	\$01	
2	\$02	
.	.	
.	.	
.	.	
190	\$BE	I/O (2K)
191	\$BF	
192	\$C0	
193	\$C1	
.	.	
.	.	
198	\$C6	I/O ROM (2K)
199	\$C7	
200	\$C8	
201	\$C9	
.	.	
.	.	
206	\$CE	ROM (12K)
207	\$CF	
208	\$D0	
209	\$D1	
.	.	
.	.	
254	\$FE	ROM (12K)
255	\$FF	

Figure 5. System Memory Map



PRINTERS

SILENTYPE™

INITIALIZATION PROCEDURES

	FROM BASIC	FROM THE MONITOR	FROM PASCAL
to initialize	PR#1	1 CTRL-P	n/a
to deactivate	PR#0	CTRL-P	n/a

PARAMETER TABLE

Function	Monitor Address	Basic Address	Coldstart Value		Possible Values
			dec	hex	
form feed	CF0D	-12531	40	\$28	from 0 (\$0) to 255 (\$FF)
line increment	CF0E	-12530	2	\$02	from 0 (\$0) to 255 (\$FF)
direction	CF0F	-12529	0	\$0	bi-directional = 0 (\$0) to 127 uni-directional = 128 to 255 (\$FF)
intensity	CF10	-12528	5	\$05	from 0 (\$0) to 7 (\$7)
left margin	CF11	-12527	2	\$02	from 0 (\$0) to 82 (\$53)--must be less than right margin
right margin	CF12	-12526	81	\$51	from 1 (\$1) to 83 (\$54)--must be greater than left margin
hi-res page	CF13	-12525	32	\$20	hi-res page 1 = 32 (\$20) hi-res page 2 = 64 (\$40)
inverse	CF14	-12524	255	\$FF	normal = 255 (\$FF) (white on black) inverse = 0 (\$0) (black on white)

CHARACTER	FUNCTION	COMMENTS
CTRL-C	stops hi-res screen print	BASIC and Monitor keyboard only
CTRL-F	prints contents of buffer	
CTRL-H	backspace	
CTRL-J	line feed	
CTRL-L	form feed	
CTRL-M	return	
CTRL-Q	prints the hi-res screen	default = hi-res page 1
CTRL-T	toggles screen echo/no echo	default = no echo

PAPER TIGER

Starting to Use Your New Printer—An Example

Once the Parallel Printer Card is configured to match the printer and installed in a slot, it is ready to use.

The Printer Commands in the examples below are in Apple II BASIC. Apple Monitor I/O Commands and Apple Basic Program I/O Commands are listed at the end of this section.

Command	Explanation
SET B ^C RETURN	Interrupts any program execution and transfers control to BASIC.
PR#1	Turns Parallel Printer Card on. All data displayed on the TV monitor screen is also sent to the printer with the following exceptions: • Graphic data is not printed • Backward TABs and VTABs do not work. (Forward TABs work correctly.)

Notes: I/O slot 0 cannot be entered.

If the incorrect slot number is typed, all output will go to the wrong slot and no data will be displayed or printed. Depress RESET, and the computer will return to TV monitor display only.

If the printer is not plugged in, turned ON, and ready to print, it will look BUSY to the system; which will "hang up" waiting for it. To clear this waiting state, get the printer ready to print (see its manual), then depress Apple II's RESET key (to clear the busy flag). The program may then be re-started.

PRINT 6 + 7 RETURN	The statement and its result should appear on both the printer and the monitor screen.
-----------------------	--

Notes: Most printers wait for a Carriage Return and then print the entire line.

Since the TV monitor display is designed for 40 characters per line, the printer will behave like a 40 column (40 characters per line) printer as long as the monitor screen display is on.

On some printers, several lines must be printed before the paper advances far enough for the first line to be seen.

At this point, data does not print at the printer:

1. Verify the Parallel Printer Card is into slot#1 (specified in the PR#1 command). Slot#1 is the **second** slot from the left.
2. Verify that the wiring of the cable and jumper block are correct.
3. Verify that PR#1 was typed correctly.
4. Verify that the printer is turned on, and that the system has been RESET since.

If the printer still does not print, ask your local Apple dealer for assistance.

Printer Commands

The Parallel Printer Card Commands begin with CTRL I (I^C). The command conventions are:

LOWER CASE

WORDS

Enter the data identified by the word.

UPPER CASE

CHARACTERS

Type the character(s) or number(s) shown.

CONTROL

CHARACTERS

Control characters are indicated by a superscript C; e.g.: I^C. A control character is entered by depressing the CTRL key and the character key simultaneously. (Similar to using the shift key to type a capital letter.)

SPACING

Spacing in the command format is for legibility only. Spaces are not required when the command is entered.

For example, I^Cno N RETURN means:

1. Type I while holding the CTRL key down.
2. Enter a number at the keyboard.
3. Type N at the keyboard.
4. Depress RETURN.

The Printer Commands are shown below. They may be used in the command mode from either BASIC or the Monitor (except for PR# and P^C commands—see explanations).

Command

Explanation

I^C n N RETURN

Turns off monitor screen and prints n columns per line on printer. The number of columns may be any number from 40 to 255.

I^C I RETURN

Returns output to TV monitor screen as well as to printer.

I^C K RETURN

Turns off automatic printer Line Feed.

I^Cletter^CRETURN

Changes printer command control character recognized by printer. For example I^C A^C changes the control character recognized by the printer.

letter ^C RETURN	Changes printer control character back to I ^C . For example, A ^C I ^C .
PR# slot no.	Turns Printer Card on from BASIC. The slot number must identify the slot in which the Parallel Printer Card is inserted. Any number from 1 to 7 may be entered.
PR#0 RETURN	Turns the Parallel Printer Card off from BASIC.
slot no. P ^C	Turns on Printer Card from the Monitor. The slot number must identify the slot in which the card is inserted.
OP ^C	Turns the Printer Card off from the Monitor. Notes: For users of Applesoft BASIC on cassette tape: Appesoft BASIC does not yet allow "PR#" commands (which are used in Apple Integer BASIC to turn the printer ON and OFF). Therefore, the printer must be controlled as follows: To turn ON, type POKE 54, 0: POKE 55, 192 + slot no. To turn OFF, type POKE 54, 240: POKE 55, 253 These commands must be entered on a single line as shown. They will work from the Monitor and Apple BASIC as well as from Applesoft BASIC, but are not required. All other commands using I ^C , work from Applesoft BASIC.

Using Printer Commands in BASIC Programs

Printer control within BASIC programs is accomplished by embedding the commands (shown above) in PRINT statements.

10 PR#1	Turns off Printer Card.
20 PRINT "I ^C no. N";	Turns off TV monitor screen display and prints n columns per line at the printer
30 PRINT "I ^C I";	Returns output to TV monitor screen as well as printer.
40 PRINT "I ^C K";	Turns off the Line Feed code.
50 PRINT "I ^C A ^C ";	Changes I ^C to A ^C for printer listing of BASIC program.
60 PRINT "A ^C I ^C ";	Restores I ^C as the printer control command character.

Example Of Control From a BASIC Program

Here is a typical BASIC program using the printer control commands.

```
10 PR #1          Turn on Printer Card.
(20 PRINT "IC K";) Only if printer advances line on Line Feed code.
30 PRINT "IC 80 N"; Output on printer only.
40 PRINT "PRINTER"
50 PRINT "IC      Output on screen and printer
60 PRINT "SCREEN AND PRINTER"
70 PR#0          Turn off printer card.
80 PRINT "SCREEN ONLY"
90 END
```

Listing Programs Containing Print Commands

To list a BASIC program containing printer control commands, take the following steps:

Command	Explanations
IPR#1	Turns Parallel Printer Card on.
I ^C K RETURN	Only if printer advances line on Line Feed code.
I ^C A ^C RETURN	Changes I ^C character to A ^C character The printer treats I ^C command sequences in a BASIC Program listing as a command and changes printer operation as specified in the command. To avoid this problem the I ^C must be changed to another character, e.g.: A ^C .
A ^C 80 N RETURN	Turns off TV monitor screen and outputs on 80 column printer.
LIST	LIST is not displayed because the monitor display is off.
A ^C I ^C RETURN	Enter after the listing is complete to restore I ^C as the printer control command character.

FIRMWARE DESCRIPTION

The commented firmware listing on the following pages fully describes the Apple II Parallel Printer Interface Card Firmware. The listing contains four sections:

- Address Transformation Information

Note: The Firmware listing provides the apparent address of the printer card to the CPU. Due to address mapping, the apparent address and the real PROM address do not agree.

- Printer Card Equates
- Printer Card Firmware Listing
- Symbol Cross - Reference Table

The PR# BASIC Command or the P^C Monitor Command are not required to access the printer board firmware.

For direct output,

- a) preset MSTRT (\$5F8 + \$N) margin start
 MODE (\$678 + \$N)
 ESCHAR (\$6F8 + \$N)
 FLAGS (\$778 + \$N) B7 = video B0 = CRLF

b) enter the program once at \$CN00. (The normal entry point is \$CN02).

Data in the accumulator is output on the data lines with STR when the responding device is ready.

The firmware program exits by an RTS or a JMP COUT1, depending on B7 of the MODE word in memory. The accumulator, X and Y registers, and stack pointer are not affected.

OPERATING HINTS

The three techniques detailed below will guard against the most common printing problems.

1. Before using the printer in your program (PR#1 statement), be sure to HOME the cursor and clear the screen. (A CALL-936 statement in your program does this.)
2. If you are printing more than 40 characters per line, be sure to re-set the line length to 40 characters per line before using the PR#0 command (which turns off the printer interface).
3. Before using the printer to list a program that has printer control commands embedded in it, change the control character from I to some other character. Then re-set I as the control character before running the program.

USING THE PRINTER CARD AS A GENERAL-PURPOSE, PARALLEL OUTPUT PORT

The Parallel Printer Card can be used as a general-purpose, 8-bit parallel output card to drive music synthesizers, digital-to-analog converters, etc. If data is stored at location \$C080 + \$N0 (where N is the slot number), then the data will appear on Printer Data Lines (DPO-DP7), and will remain until the next "STORE" instruction to that location is executed.

From BASIC this data transfer may be accomplished by typing:

POKE (—16256 + N16), DATA

N is the slot number of the Printer Card, and DATA is the Data to be put out.

Each time a byte is sent to the Printer Card, a strobe will be generated on the STR line. The strobe polarity may be set as described earlier for strobes printers.

PAPER TIGER 440G

DIP S 4

60/60 HZ SELECT. SET FOR
POWER SOURCE FREQUENCY

SW7

OFF = 60 HZ
ON = 50 HZ

REMOTE CONTROL DESELECT.
ENABLE/DISABLE RECOGNITION OF DC1 AND DC3 ASCII

SW6

OFF = DISABLE
ON = ENABLE

AUTOMATIC LINE FEED.
ENABLE/DISABLE A LINE FEED
UPON RECEIVING A
CARRIAGE RETURN (CR).

SW5

OFF = DISABLE
ON = ENABLE

AUTOMATIC LINE FEED.
ENABLE/DISABLE A 1-INCH
SKIP AT FORM BOUNDARIES

SW4

OFF = DISABLE
ON = ENABLE

LINE PER INCH. SELECT
PRINTING AT EITHER 6 OR
8 lpi.

SW3

OFF = DISABLE
ON = ENABLE

CHARACTER DENSITY SELECT
ONE OF FOUR CHARACTER
PRINT DENSITIES.

SW1 SW2

OFF OFF = 8.3 cpi
OFF ON = 12 cpi
ON OFF = 10 cpi
ON ON = 16.5 cpi

SW7

SW6

SW5

SW4

SW3

SW1/SW2

DIP S 3

SW1/SW2/SW3

PAGE SIZE SELECT ONE OF EIGHT
PAGE LENGTHS:

SW1 SW2 SW3

OFF OFF OFF = 3 INCHES
OFF OFF ON = 4 INCHES
OFF ON OFF = 3.5 INCHES
OFF ON ON = 5.5 INCHES
ON OFF OFF = 7 INCHES
ON OFF ON = 11 INCHES
ON ON OFF = 8.5 INCHES
ON ON ON = 14 INCHES

SW4/SW5

SERIAL DATA BAUD RATE.
SELECT DATA RATE FOR
SERIAL DIGITAL INPUT DATA.

SW4 SW5

OFF OFF = 110 BAUD
OFF ON = 300 BAUD
ON OFF = 600 BAUD
ON ON = 1200 BAUD

SW6

CONTROL CODES
RECEIVE. ENABLE/-
DISABLE RECONITION OF
ASCII CONTROL CODES
WHICH SELECT EN-
HANCED PRINT MODE,
OPTIONAL GRAPHICS
MODE, AND CHARACTER
DENSITY CHANGES.

SW6

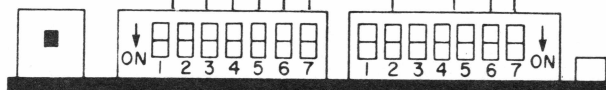
OFF = DISABLE
ON = ENABLE

SW7

SERIAL/PARALLEL IN-
TERFACE SELECT. SET
FOR TYPE OF INTER-
FACE TO DATA
SOURCE.

SW7

OFF = PARALLEL TTL
ON = SERIAL EIA



DIP S 4 (TOP VIEW) DIP S 3

PAPER TIGER 460G

DIP S 4

SW7

EXPANDED FUNCTION
(SOFTWARE CONTROL)

SW7

OFF = INHIBIT FUNCTIONS
ON = ENABLE EXPANDED
FUNCTIONS

FIXED/PROPORTIONAL
SPACING

SW6

OFF = FIXED SPACING
ON = PROPORTIONAL
SPACING

AUTOMATIC LINE FEED.

ENABLE/DISCHARGE A LINE
FEED UPON RECEIVING A
CARRIAGE RETURN (CR).

SW5

OFF = DISABLE
ON = ENABLE

AUTOMATIC PAGE SKIP.

ENABLE/DISABLE A 1-INCH
SKIP AT FORM BOUNDARIES

SW4

OFF = DISABLE
ON = ENABLE

LINES PER INCH. SELECT
PRINTING AT EITHER 6 or
8 lpi.

SW3

OFF = 6 lpi
ON = 8 lpi

SW1/SW2

CHARACTER DENSITY SELECT
ONE OF FOUR CHARACTER
PRINT DENSITIES

SW1 SW2

OFF OFF = 5 cpi
(ENHANCED)
OFF ON = 12 cpi
ON OFF = 10 cpi
ON ON = 16.5 cpi

DIP S 3

SW1/SW2/SW3

PAGE SIZE SELECT ONE OF EIGHT
PAGE LENGTHS:

SW1 SW2 SW3

OFF OFF OFF = 3 INCHES
OFF OFF ON = 4 INCHES
OFF ON OFF = 3.5 INCHES
ON OFF OFF = 7 INCHES
ON OFF ON = 11 INCHES
ON ON OFF = 8.5 INCHES
ON ON ON = 14 INCHES

SW4/SW5

SERIAL DATA BAUD RATE.
SELECT DATA RATE FOR
SERIAL DIGITAL INPUT DATA.

SW4 SW5

OFF OFF = 1200 BAUD
OFF ON = 300 BAUD
ON OFF = 600 BAUD
ON ON = 1200 BAUD

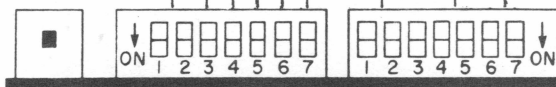
SW6/SW7

PARITY SELECT

SW6 SW7

OFF OFF = NO PARITY
CHECK TRANSMIT
SPACE PARITY
ON OFF = NO PARITY
CHECK TRANSMIT
MARK PARITY
OFF ON = PARITY
CHECK TRANSMIT
EVEN PARITY
ON ON = PARITY
CHECK TRANSMIT
ODD PARITY

NOTE: ALWAYS RE-
MOVE POWER FROM
THE PRINTER PRIOR TO
SETTING OR RESETTNG
DIP SWITCHES.



DIP S 4 (TOP VIEW)

DIP S 3

SPINWRITER™ PROGRAM CARDS

ABSOLUTE HORIZONTAL TAB

Table 1

1st	ESC															
2nd	P				Q				R				S			
3rd	@	P	17	@	P	17	@	P	17	@	P	17	@	P	17	@
A	1	17	33	49	65	81	97	113	129	145	161					
B	2	18	34	50	66	82	98	114	130	146	162					
C	3	19	35	51	67	83	99	115	131	147	163					
D	4	20	36	52	68	84	100	116	132	148						
E	5	21	37	53	69	85	101	117	133	149						
F	6	22	38	54	70	86	102	118	134	150						
G	7	23	39	55	71	87	103	119	135	151						
H	8	24	40	56	72	88	104	120	136	152						
I	9	25	41	57	73	89	105	121	137	153						
J	10	26	42	58	74	90	106	122	138	154						
K	11	27	43	59	75	91	107	123	139	155						
L	12	28	44	60	76	92	108	124	140	156						
M	13	29	45	61	77	93	109	125	141	157						
N	14	30	46	62	78	94	110	126	142	158						
O	15	31	47	63	79	95	111	127	143	159						
	16	32	48	64	80	96	112	128	144	160						

ASCII Code  Column Address

ESCAPE SEQUENCES (TWO LEVEL)

ESC 1	Set Horizontal Tab
ESC 2	Reset Horizontal Tab (Individual)
ESC 3	Print in Red
ESC 4	Print in Black
ESC 5	Set Vertical Tab
ESC 6	Reset Vertical Tab (Individual)
ESC 7	Clear all Tabs and FF Length
ESC 9	Reverse Line Feed
ESC <	Reverse Print (Right-to-Left) On
ESC =	Read and Store Operator Control Switches
ESC >	Forward Print (Left-to-Right) On
ESC ?	Set Format Mode
ESC @ CR'	Reset Format Mode
ESC J or j	Set Right Margin
ESC K or k	Reset Right Margin
ESC L or l	Set FF Length
ESC M or m	Set Left Margin
ESC O or o	Reset Left Margin

RELATIVE VERTICAL TAB

Table 2

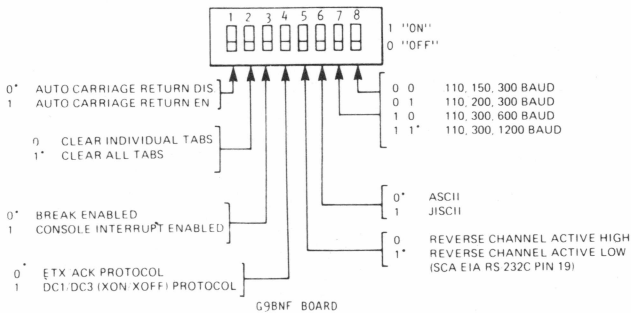
	Reverse				Forward			
	ESC				ESC			
1st								
2nd	X		Y		Z		I	
3rd	0	16	32	48	0	16	32	48
A	1	17	33	49	A	1	17	49
B	2	18	34	50	B	2	18	50
C	3	19	35	51	C	3	19	51
D	4	20	36	52	D	4	20	52
E	5	21	37	53	E	5	21	53
F	6	22	38	54	F	6	22	54
G	7	23	39	55	G	7	23	55
H	8	24	40	56	H	8	24	56
I	9	25	41	57	I	9	25	57
J	10	26	42	58	J	10	26	58
K	11	27	43	59	K	11	27	59
L	12	28	44	60	L	12	28	60
M	13	29	45	61	M	13	29	61
N	14	30	46	62	N	14	30	62
O	15	31	47	63	O	15	31	63

ASCII Code



Number of lines

5510 AND 5520 SERIAL I/F CONFIGURATIONS OPERATOR CONSOLE

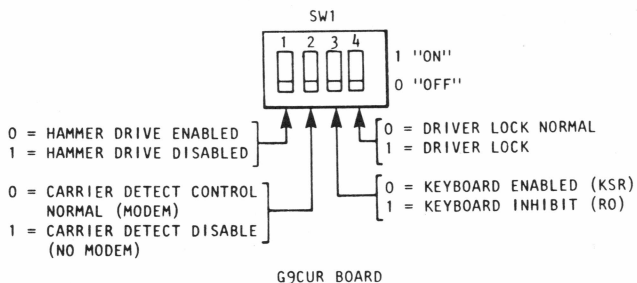


*INDICATES STANDARD CONFIGURATIONS

SPACING CONTROL AND FORM ADVANCE CONTROL

Table 3

		Spacing		Form Advance	
1st	2nd	3rd	(inches)	3rd	(inches)
ESC	]	@	0	P	1/48
		A	1/120	Q	2/48
		B	2/120	R	3/48
		C	3/120	S	4/48
		D	4/120	T	5/48
		E	5/120	U	6/48 (1/8)
		F	6/120	V	7/48
		G	7/120	W	8/48 (1/6)
		H	8/120	X	9/48
		I	9/120	Y	10/48
		J	10/120 (1/12)	Z	11/48
		K	11/120	[	12/48
		L	12/120 (1/10)	\	13/48
		M	13/120	]	14/48
		N	14/120	^	15/48
		O	15/120	—	16/48



ASCII CODING CHART

					COMMUNICATION CODES		PRINTABLE CHARACTERS						
Bits	b7				0	0	0	0	1	1	1	1	
	b6				0	0	1	1	0	1	0	1	
	b5				0	1	0	1	0	1	0	1	
	b4	b3	b2	b1	Column →								
	↓	↓	↓	↓	Row ↓	0	1	2'	3	4	5	6	7
	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
	0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
	1	0	0	1	9	HT	EM	)	9	I	Y	i	y
	1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
	1	0	1	1	11	VT	ESC	+	; ,	K	[	k	}
	1	1	0	0	12	FF	FS	,	<	L	\	l	:
	1	1	0	1	13	CR	GS	=	=	M		m	}
	1	1	1	0	14	SO	RS	.	>	N	^	n	~
	1	1	1	1	15	SI	US	/	? ,	O	_	o	DEL

Note

Both column 4 ~ 5 (capital letter) and column 6 ~ 7 (small letter) in ASCII code table of all ESC code sequences have same function.

(Except DEL code)

CONTROL CODES

Signal	Name
ETX	End of Text
ACK	Acknowledge
BEL	Bell
BS	Backspace
HT	Horizontal Tab
LF	Line Feed
VT	Vertical Tab
FF	Form Feed
CR	Carriage Return
SO	Shift Out
SI	Shift In
DC1	Device Control 1
DC3	Device Control 3
ESC	Escape

**HARDWARE
CARDS**

SPECIAL ROMS AVAILABLE FOR MOUNTAIN HARDWARE ROM+ BOARD

The following ROM Programs are currently available from Soft CTRL Systems* at \$35.95 each and are very helpful in programming and program operation.

DISK COPY/DISK SPACE IN ROM

Putting DISK COPY into operation

Disk Copy can be selected either from Applesoft or Integer Basic.

To Access a ROM Program on the Mountain Hardware Board:

- A) PR#n (C/R), where n is the slot number that your ROMPLUS board is in.
- B) The following keys are all pressed simultaneously; the CTRL key, the SHIFT key, and the N key. Release the three keys and type the number of the socket in which the ROM is installed, then type the 'A' or 'B' key (As specified in the ROM Program)
- C) Right after the 'A' or 'B' key is pressed, DISK COPY will be up and running.

Use 'A' to select DISK COPY will prompt for all required inputs. See 'Explanation for prompt questions' for further information.

Putting DISK SPACE into operation

1. Follow the same instructions as for DISK COPY except type the 'B' key in place of the A key.
2. Right after the B key is pressed, DISK SPACE will clear the monitor screen, read the disk which is in the drive that was last activated, and display on the monitor, the actual number of free sectors left on the disk. The number of kilobytes displayed in parenthesis is approximate and will be within plus or minus 1K. NOTE: If you are using DISK SPACE on a 32K Apple, any Integer program in memory will be clobbered.

When either DISK COPY or DISK SPACE is activated, the ROMPLUS board shuts down when the programs are executed and needs no further attention.

NOTE: Disk Copy and Disk Space, when activated, will recognize which DOS is up (13 or 16) and operate accordingly. However, if, as an example, 3.3 (16 sector) DOS is up and an attempt is made to read a 13 sector disk, both Disk Copy and Disk Space will abort the read (or write).

*Soft CTRL Systems
Box 599, West Milford, N.J. 07480

EXPLANATION OF PROMPT QUESTIONS:

OVERWRITE DOS ?

If you answer yes 'Y', DISK COPY will use another 7.5K of RAM, which is presently occupied by DOS. This will increase the number of sectors that can be read into memory. The result is a shorter copy time.

GROSS COPY ?

If you answer yes 'Y', DISK COPY will copy every sector on the disk whether the sector is in use or not. This option should be selected when it is known that the master or source disk does not have a Volume Table of Contents or the master is a data only, disk. Pressing any other key will by-pass the Gross Copy routine and go to the Disk Content Copy routine. This copy routine will copy only those sectors which are active or in use and should be selected when a Volume Table of Contents exists on the master or source disk. (The disk is not data only.) This routine is the quicker of the two options.

NOTE: If you answer no to the gross copy question, Disk Copy will make an attempt to abort if a master disk does not have a Volume Table of Contents on it.

DEFAULT VALUES FOR SLOT AND DRIVE NO.

MASTER AND COPY DISKETTE, SLOT = 6

MASTER AND COPY DISKETTE, DRIVE = 1

DISK COPY will present these default values, one at a time, and wait for an input response. A Return key entry will leave the default value or you can change them to suit your system.

INITIALIZE COPY DISKETTE ?

If you answer yes 'Y', DISK COPY will request a volume number for the copy disk.

NOTE: If the master or source disk is a slave disk (one that has not been UPDATE'D or MASTER CREATE'D) and is copied onto a disk initialized with a different volume number, the copy disk will no longer boot. However, the copy disk can be UPDATE'D or MASTER CREATE'D which will then allow it to be booted.

After a volume number is entered, DISK COPY formats the copy diskette. However, DOS will not be put on the copy diskette until the copy routine is started.

ABORT ?

If DISK COPY encounters any one of the following, it will request an answer to the abort question;

DRIVE ERROR

NOTE: On 13 sector disks, drive errors can be expected on track #2- sectors 10, 11, 12, and 13. It is suggested you respond with a not 'N' answer to the abort question under this circumstance.

WRITE ERROR

VOLUME MISMATCH

READ ERROR

If you answer yes 'Y', DISK COPY will terminate the present copy routine and request an answer to the question;

ANOTHER COPY ?

A yes 'Y' answer will reset DISK COPY for a new copy and present the question at the completion of a successful copy. If you answer no 'N', and you answered yes to the overwrite DOS question, DISK COPY will request an answer to the question, REBOOT THE DISK ? You can, at this time, place the copy diskette in the master diskette drive and answer yes 'Y' to this question. If you answer no 'N', DISK COPY will return you to basic and will leave you with the following statement; DOS IS NOT UP!

REBOOT THE DISK ?

If you answer yes 'Y', DISK COPY will boot the diskette which is in the drive that has the master diskette. A partial test of a copy disk is to see if it will boot. If you insert the copy diskette into the drive which has the master diskette, DISK COPY will boot it.

CATALOG CMND MENU / DISK MAP IN ROM

1. Make sure DOS 3.2, 3.2.1, or 3.3 is in operation.
2. The CMND ROM can be activated from either Applesoft, Integer Basic or from the System Monitor. Use the 'A' key to select.

After the 'A' key is pressed the CMND ROM will catalog the diskette which is in the drive that was last used and a menu will be put at the top of the screen. Below the menu, in inverse video, the volume number of the diskette will be displayed; along with the total number of pages assigned to hold the catalog directory listing, the number of the page you are presently looking at, the number of current disk drive, and the number of sectors in use and the number of sectors left for file storage.

A WORD ABOUT THE NUMBER OF USABLE SECTORS ON A DISKETTE:

On a 13 sector diskette there is 13 sectors per track and there is 35 tracks. The total number of sectors is therefore, 455. For a 16 sector diskette, the total number of sectors is 560. However, on both a 13 and 16 sector diskette, DOS is contained on three tracks and the Volume Table Of Contents is contained on another track. These four tracks are not normally available for your use and reduces the total number of usable sectors to 403 for a 13 sector diskette, and 496 for a 16 sector diskette. If you add the number of sectors used and the number of usable sectors to 403 for a 13 sector diskette, and 496 for a 16 sector diskette. If you add the number of sectors used and the number of sectors left you will arrive at one of these numbers, depending on which size diskette you are currently using.

Below the statistical data on the screen, concerning the current diskette, the first page of file names, found in the disk directory, is displayed. Each page will hold up to 16 file names. Each file name has a letter assigned to it for identification and access. If the space bar is pressed at this time, the next page of sixteen file names (If there is that many) will be

displayed along with the page number you are looking at. The catalog directory on a 13 sector diskette will hold up to 84 file names and 105 file names on a 16 sector diskette. This means there can be up to 6 pages of file names for a 13 sector diskette and up to 7 pages for a 16 sector diskette.

COMMAND KEYS:

CMND ROM will recognize command letter keys only. Any other keys pressed will beep the speaker. The command menu, on the screen, identifies each command key and its function. Each command letter is followed by a right parenthesis and their functions are as follows:

L Will put the word 'LOAD' at the top of the screen and wait for you to press one of the letter keys assigned to a file. If the file is an 'A' or 'I', the screen will clear, the CMND ROM will be shut down (including the Romplus Board), the file will be loaded and you return to the basic language of the file.

If a 'B' file is loaded, the start and length addresses will be displayed (In decimal form). The addresses remain on the screen for approximately 6 seconds, and allow return to the command menu.

R Will put the word 'RUN' on the screen and wait for a corresponding file letter key to be pressed. The CMND ROM will shut itself down and the file will be loaded and run.

THE CMND ROM WILL IGNORE AN ATTEMPT TO RUN OR LOAD A TEXT FILE.

C Read, retain, and display the first page of the disk catalog directory of the diskette in the current drive.

D Delete a file - with verify. The verify command will request a 'Y' or 'N' key press before a file deletion will take place. If you try to delete a file that is locked, a 'FILE LOCKED' message will appear and return to the command menu.

K Lock a file or all files. To lock all files on diskette, follow with an 'X' command letter. Each page of listings and file is displayed as it is being locked.

U Unlock a file or all files. To unlock all files, follow with an 'X' command letter. Each page and file is displayed as it is being unlocked.

M Display a track / sector map of the current or most recently cataloged diskette. Locations of used and unused sectors are shown in relation to each other, all track and sector numbers are labeled in hexadecimal notation.

T Change and display current drive number. Change the drive by entering the 'T' command letter, CMND ROM will consult with DOS to see if two drives exist. If so, the new drive number will be displayed and the diskette in the drive will be read.

E Exit - Clears the screen and returns to the current Basic. Both CMND ROM and ROMPLUS BOARD will be shut down. If the last file was a blood you return to the System Monitor.

SPACE BAR - Display the next page of the file listings without recataloging the diskette. The space bar may also be used to cancel a command letter entry.

THE RESET KEY

If your APPLE has the 'AUTO START ROM', the CMND ROM will not allow you to do a violent exit by pressing the reset key. Two beep, and 'I/O ERROR' message and the command will appear if tried. To exit press the 'E' command letter.

If your APPLE does not have the 'AUTO START ROM' and the reset key is pressed, the system monitor and CMND ROM will be disabled. Enter the following to correct:

3D0G C/R

OR

3D3G C/R

This reconnects DOS and returns to the current Basic language. An exit from CMND ROM will insure that DOS remains intact and operational.

APPLESOFT RENUMBER / MERGE IN ROM

Putting RENUMBER/MERGE IN ROM in operation.

1. Make sure your computer is in Applesoft basic. Use the 'A' key.
2. Right after the 'A' key is pressed you will see Applesoft's prompt character appear on the next line of your monitor screen.
3. The RENUMBER/MERGE program is now in operation and waiting for your commands. Follow the operating syntax as outlined in the instructions you received with the Renumber/Merge program from Apple Computer, Inc. on the DOS system master diskette.
4. To free up the memory occupied by the RENUMBER/MERGE program, execute and 'FP' command or reset HIMEM: to its normal setting. Check your DOS manual if your using a disk drive or Applesoft manual if your not using a disk drive.
5. The ROMPLUS board, which was turned on by your keyboard commands, has been shut down by the RENUMBER/MERGE program and needs no further attention.

'BASICS' IN ROM

Putting 'BASICS' IN ROM into operation.

Cold start- With out the Autostart ROM:

When the Apple is first turned on- you will be in the monitor. At this time do one of two things; address the Basics ROM from the monitor or go to one of the basics with a CTRL B and address it.

From the monitor:

1. Place a 13 sector diskette in the Disk II drive-I.
2. Enter, from the keyboard the following direct command;
 - A) Press the numerical key which is equal to the number of the slot that your Romplus board is in. Release this key and while holding

down the CTRL key press the 'P' key. Release both keys and press the Return key.

- B) The following keys are all pressed simultaneously; the CTRL key, the SHIFT key, and the 'N' key. Release the three keys and type the number of the socket in which the 'BASICS' ROM is installed, then type the 'A' key.
- C) Right after the 'A' key is pressed the screen will clear and you will see;

SLOT # ?

At this time, enter the slot number you wish to boot from. The default slot is six. Press the Return Key in place of a numerical key, the Basics ROM boots the diskette in the default slot.

3. The 13 sector diskette, after it has booted, runs the greeting program listed in DOS. Disk II and DOS now recognize only 13 sector diskettes.

From Basic- Applesoft or Integer:

Follow all steps listed at the front of this section.

Cold start- With the Autostart ROM:

After the Autostart ROM has been booted a 16 sector disk, address the Basics ROM as outlined under cold start without the Autostart ROM. If you use firmware cards, either Applesoft or Integer, and the Autostart ROM is on it, you can deselect the Autostart ROM by flipping the toggle switch on the card. This should be done before you turn the Apple on. When the Apple is turned on, it will not boot a disk in the drive and avoids an unnecessary step. Now follow the steps listed for Cold start without the Autostart ROM.

Warm start- With or without the Autostart ROM:

The Basics ROM can be addressed at any time the Apple is on. Follow the same syntax as listed for Cold starts.

LIST CTRL IN ROM

Putting LIST CTRL IN ROM into operation.

1. Make sure your computer is in Applesoft basic and DOS is up.
2. Enter, from the keyboard, the direct commands and use the 'A' key.

Right after the 'A' key is pressed the screen clears and LIST appears in the top left corner of the screen and waits for any one of three inputs.

1—An ESC key press

2—A RETURN key press

3—Line number or numbers of the program presently in memory and a RETURN key press. (Use proper Applesoft syntax for entering line numbers.)

If the ESC key is pressed at this point, LIST CTRL will present the question:

WILL YOU NEED LIST CTRL AGAIN ? (Y/N)

A press of the 'Y' key exits you from LIST CTRL and allows you to call it back by entering the Ampersand character (&) followed by a RETURN key press.

Any program you may have in memory at the time you address the LIST CTRL ROM will not be disturbed in any way by LIST CTRL.

Upon entry of a RETURN key press or line numbers of your program, followed by a RETURN key press, LIST CTRL will then start to listout your program.

The setting of paddle (0) controls a time delay period of between zero and two and three quarter (0-2¾) seconds between each line listed to the screen. The setting of paddle (1) controls a time delay period of between zero and three quarters of a second (0-¾) between each character outputted to the screen. (Similar to the Applesoft SPEED command) These paddle settings can be changed at anytime during, before, and after a listout, in order to suit a comfortable reading speed. At any time during the listout, either paddle button can be utilized to stop and start the listout. This allows time to study or take notes and then continue with the program listing. When a listout is stopped a beep sounds from the speaker and the PAGE AT A TIME feature enabled. Pressing the spacebar at this time, clears the screen, and the next page of text (20 lines) is listed on the screen. Each succeeding press of the spacebar clears the screen and presents the next page of the program listing. Pressing either paddle button after a page listout beeps the speaker again, disables the PAGE AT A TIME feature, and continues the listout at the line/character output rate, set by the paddles.

LIST CTRL can be interrupted at any time by pressing the ESC key. This will abort any listout to the screen and request an answer to the following question;

ANOTHER LIST ? (Y/N)

If a Y answer is given, the screen clears and LIST appears in the top left corner of the screen and waits for input of another listout command.

If an N answer is given, LIST CTRL will request an answer to the following question;

WILL YOU NEED LIST CTRL AGAIN ? (Y/N)

If a Y answer is given, LIST CTRL resets your systems HIMEM: to protect itself. This allows your programs to run without clobbering LIST CTRL. (That is, provided you or your programs don't change HIMEM:.) The ampersand key also sets up with the reenter address, and allows you to call LIST CTRL by entering the ampersand character (&) and a RETURN key press.

If an N answer is given, LIST CTRL resets your systems himem: and the ampersand key address to their original settings and free up the .8K of memory used by LIST CTRL.

These same questions will also be presented when a listout has been completed. A completed listout will be recognized when the Applesoft prompt character is displayed above the question on the screen, ANOTHER LIST ? (Y/N).

If the ESC key is pressed while the word LIST is on the screen and waiting for your input, the first question will not be asked. This allows for a little quicker exit from LIST CTRL.

If the RESET key is pressed at any time while LIST CTRL is active or on call, it is suggested you address the LIST CTRL ROM again. (After you have returned to Applesoft basic if you don't have the AUTO-START ROM)

The ROMPLUS board, which was turned on by your keyboard commands, has been shut down by the LIST CTRL ROM and needs no further attention.

REFERENCE

APPLESOFT

Random number generator isn't random (patterns)

Enclosed is a copy of an applications note describing Applesoft's random number generator.

The Applesoft random number generator, like all such routines, is only a pseudo-random generator, so non-random patterns will eventually occur. The frequency of repetition of these patterns will vary from procedure to procedure. Proper re-seeding of the random number generator during a program will help prevent the appearance of large repeating sequences. This can be done in two ways, and for best results, both should be used.

1. Seed the random number calculation at the beginning of the program, using the keyboard count location. This will take the form

$$S = \text{PEEK}(78) + \text{PEEK}(79) * 256$$
$$X = \text{RND}(-S)$$

2. Within the random generating portion of the program, insert a statement of the form $Z = \text{RND}(-\text{RND}(9))$, which will begin a new random sequence.

Please be aware that no method will completely eliminate patterns in the random numbers generated, but we can break up the sequences so that objectional non-random patterns are less likely to appear.

Round FP numbers & set decimals

There is a formula to round Applesoft real numbers on page 18 of the Applesoft Reference manual and a program to limit the number of digits to the right of the decimal point on page 22. In addition I am including a listing of a program to format real number output.

LIST

```
100      REM          PRINT USING
110      REM
120      REM          SIMULATOR
130      REM
140      REM          AUG 79
150      REM
160      REM          J. CROSSLEY
170      REM
180      LET N = 2: REM SET NUMBER
190      REM          OF DECIMALS
200      LET S = 5: REM SET FIELD
210      REM          WIDTH
220      HOME
230      FOR X = - 5 TO 5 STEP .501
240      PRINT X, "$";
250      GOSUB 2000
260      PRINT
```

```

270      NEX X
280      PRINT
290      PRINT "UNFORMATTED  FORMATTED"
300      END
1000     REM THIS IS THE FORMATTING
1010     REM SUBROUTINE. THE INPUT
1020     REM IS 'X','N', AND 'S'
1030     REM X IS THE NUMBER TO BE
1040     REM BE PRINTED
1050     REM N IS THE NUMBER OF
1060     REM DIGITS RIGHT OF '.'
1070     REM S IS THE WIDTH OF THE
1080     REM RIGHT JUSTIFIED
1090     REM PRINTING FIELD
1100     REM
2000     X$ = " " + STR$(INT(X * 10 ^ N + .5))
2010     Q = LEN(X$) - (VAL(X$) < 0)
2020     PRINT SPC(S - Q * (Q > N + 1) - (N + 2) *
      (Q <= N + 1));
2030     PRINT MID$(X$,1 + (VAL(X$) < 0),(Q <= N) +
      (Q - N) * (Q > N))
2040     PRINT MID$("0.00",1 + ((N + 1) < Q),1 +
      (N - Q + 2) * (Q < N + 2));
2050     PRINT RIGHT$(X$,N * (Q > N) + (Q - 1) * (Q <= N));
2060     RETURN

```

"PRINT USING" simulator

Applesoft doesn't contain a PRINT USING statement. However I am including a listing of a program that aligns decimal points in a printout.

Relocate Applesoft program

The easiest way to use HIRES graphics with a long Applesoft program is to relocate the program. Do the following before loading your program or as a separate loader program:

For HGR	For HGR and HGR 2
POKE 103,1	POKE 103,1
POKE 104,64	POKE 104,96
POKE 16384,0	POKE 24576,0
LOAD (your program)	LOAD (your program)

These instructions protect HIRES memory by starting the Applesoft program above it.

Integer to Applesoft program conversion

Page contains a routine which can be used to convert Integer Basic programs to Applesoft using the disk system. Integer Basic is required for the first step of the conversion only.

First, load in the Integer program. Add the routine on page at the very beginning of the program, adjust the LIST line for the specific program, and give the textfile a unique name. Then type "RUN". This will cause the program to be listed out to the disk as a textfile under the name specified. At this point, the text file is independent of the language.

To convert, type "EXEC < your filename > " in Applesoft. The textfile will be read into the system as if typed from the keyboard. Since Applesoft does not check syntax at input time, the Integer syntax is accepted, although the program may not run. Save the program in Applesoft as usual.

Micro Magazine, January 1980, contains an article describing Integer to Applesoft conversion using cassette instead of disk. The article also describes some of the syntax and logic changes that will usually have to be made.

Tones don't work

The tones subroutine in the old Apple II Reference manual (the Red Book) will not work with Applesoft because it uses memory needed by Applesoft. I am including a copy of the tones subroutine that has been relocated to a safe area in memory.

LIST

```

10      REM *****
20      REM *
30      REM *          SIMPLE TONES FOR          *
40      REM *
50      REM *          APPLESOFT II              *
60      REM *
70      REM *          J CROSSLEY                *
80      REM *
90      REM *****
100     REM
110     REM THIS IS THE ROUTINE
120     REM FROM PAGE 45 OF THE
130     REM 'RED BOOK' MODIFIED
140     REM FOR APPLESOFT.
150     REM REM
160     REM HOME
170     REM LIST - 150
180     REM
190     REM INSERT THE FOLLOWING
200     REM ROUTINE IN YOUR PROGRAM
210     REM THEN POKE 768,TONE
220     REM      POKE 769,DURATION
230     REM AND, CALL 770 FOR TONES
240     REM

```

```

250      FOR I = 770 TO 790
260      :: READ J
270      :: POKE I,J
280      NEXT
290      DATA 173,48192,136,208,5
300      DATA 206,1,3,240,9,202,208
310      DATA 245,174,0,3,76,2,3,96
320      FOR TN = 100 TO 10 STEP - 1
330      :: POKE 768,TN: REM TONE
340      :: POKE 769,10: REM DURATION
350      :: CALL 770
360      NEXT
370      HOME
380      LIST 180,360

```

Use FRE(0) for garbage collection

Applesoft stores its strings variables dynamically. However when it runs out of free memory it must clean up the unused strings, a process which can take several minutes. There is no way to stop this process but you can give your program's user some warning. The following line should be put in the main flow of your program.

```
1000 IF PEEK(112)-PEEK(110) 4 THEN PRINT"STANDBY"; A = FRE(0)
```

FRE(0) gives a negative number

Sometimes PRINT FRE(0) prints a negative number. This will only happen in a 48K Apple and is caused by the way Applesoft handles integer numbers. Any number greater than 32767 will appear to have 65536 subtracted from it. So just add 65536 tot he value returned. This can be done from a program like this:

```
1000 PRINT FRE(0) - 65536 * (FRE(0) 0)
```

DOS

Crashed disk recovery

There are three basic ways for the information on your diskette to become damaged:

- 1) Destruction of data - through damage to media or accidental erasure.
- 2) Damage to the Directory information or the Volume Table of Contents (VTOC).
- 3) Damage to the DOS itself.

Lets take these in reverse order . . .

If the DOS is damaged you will usually find that the diskette will not boot but that programs and files can be accessed by booting another (undamaged) diskette, inserting the first one and proceeding normally from there. You can sometimes recover from this problem by running the UPDATE 3.2.1 program on the offending diskette. If this doesn't work you will have to transfer the files one-by-one to a newly initialized diskette.

Damage to the directory or VTOC is a little more complex but still (usually) recoverable. To do this you must obtain one of the disk utility programs currently on the market and physically repair the damage. You will find Appendix C of the DOS 3.2 Reference Manual especially helpful.

If the data itself is damaged or the diskette partially erased you will, in all likelihood, not be able to recover. Make frequent backup copies of your work to guard against this possibility.

Software slot finder

DOS 3.2 and 3.2.1 both keep a table of the current system parameters (DOS 3.3 does too). Here are the addresses of the most useful for a 48K Apple. Subtract 16384 for a 32K Apple.

CURRENT VOLUME	= PEEK (43622)
CURRENT DRIVE	= PEEK (43624)
CURRENT SLOT	= PEEK (43626)
CURRENT RECORD LENGTH	= PEEK (43628)
CURRENT RECORD NUMBER	= PEEK (43630)
CURRENT BYTE IN RECORD	= PEEK (43632)

BLOAD A,L parameters

Immediately after BLOADing a program you can find the A and L parameters with which it was BSAVED by using the pointers explained on page of the manual.

Direct vs Indirect commands

DOS 3.2 and 3.2.1 check to see if the Applesoft program is running before it will execute certain commands. If you try to restart an Applesoft program with CONT or GOTO, one of the "running" flags may not be set. So instead type POKE51,0:CONT, POKE51,0:GOTO 100, or have the POKE51,0 in the program before the next DOS command.

APPEND overwrites start of file

Enclosed is information concerning APPEND in DOS 3.2 and 3.2.1.

Yes! There Is a Fix for APPEND in DOS 3.2 (and 3.2.1)!

The problem with APPEND in DOS 3.2 is that DOS doesn't write an End Of File marker on the disk when you close a file. DOS normally fills new sectors with EOF markers, so the newly APPENDED information usually has an EOF after the last character. However, when the last character of the file falls exactly at the end of a sector, DOS doesn't find a new sector to fill with EOF markers. The next time DOS does an APPEND it can't find the EOF marker and defaults back to the beginning of the file.

The fix is to write out an EOF marker before closing the file after each write. Here is a five byte routine that will supply an EOF. It can be moved to any address if you are already using 768 to 772.


```

10      LET D$ = CHR$ (4)
20      POKE 768,169
30      POKE 769,0
40      POKE 770,76
50      POKE 771,237
60      POKE 772,253
70      REM NOW TO USE IT!
80      PRINT D$;"APPEND FILE"
90      PRINT D$;"WRITE FILE"
100     PRINT "THIS IS DATA"
110     PRINT "SO IS THIS"
120     CALL 768: PRINT : REM THIS IS IT!
130     PRINT D$;"CLOSE FILE"
140     END

```

Using this method, one need never worry about APPEND overwriting the start of a file.

Reset re-boots the disk

Booting with DOS 3.1 in a system with an Auto-Start ROM will cause a re-boot whenever reset is pressed. To prevent this you can either update the diskette to DOS 3.2.1 (see the DOS manual) or include the following in your HELLO program:

```

      POKE 1010,208
      POKE 1011,3
      POKE 1012,166

```

DOS commands and Applesoft GET

The control-D of a DOS command must be the first character on a output line. This means that it must be immediately preceeded by a carriage return. If it isn't, the command will be printed out to the screen or the printer and ignored by DOS. The most common cause for this is having a GET statement or a PRINT statement that ends with a semi-colon or comma. Another PRINT with no semi-colon or comma will generate the carriage return required for the next DOS command to be recognized.

MINI-ASSEMBLER

How to use

The Apple II mini-assembler is programming aid aimed at reducing the amount of time required to convert a handwritten program to object code. The mini-assembler is basically a look-up table for opcoded. With it, you can type mnemonics with their absolute addresses, and the assembler will convert it to the correct object code and store it in memory.

Typing "F666G" will put the user in mini-assembler mode. While in this mode, any line typed in will be interpreted as an assembly language instruction, assembled, and stored in binary form unless the first character on the command line is a "S".

If it is, the remainder of the line will be interpreted as a normal monitor command, executed, and control returned to assembler mode. To get out of the assembler mode, reset must be pushed.

If the first character on the line is blank, the assembled instruction will be stored starting at the address immediately following the previously assembled instruction. If the first character is nonblank (and not "S"), the line is assumed to contain an assembly language instruction preceded by the instruction address (a hex number followed by a ":"). In either case, the instruction will be retyped over the line just entered in disassembler format to provide a visual check of what has been assembled. The counter that keeps track of where the next instruction will be stored is the pseudo PC (Program Counter) and it can be changed by many monitor commands (eg. 'L', 'T', . . .). Therefore, it is advisable to use the explicit instruction address mode after every monitor command and, of course, when the Tiny assembler is first entered.

Errors (unrecognized mnemonic, illegal format, etc.) are signalled by a "beep" and a carrot (" ^ ") will be printed beneath the last character read from the input line by the mini assembler.

The mnemonics and formats accepted by the mini assembler are the same as those listed by the 6502 Programmers Manual, with the following exceptions and differences:

1. All imbedded blanks are ignored, except inside addresses.
2. All addresses typed in are assumed to be in hex (rather than decimal or symbolic). A preceding "S" (indicating hex rather than decimal or symbolic) is therefore optional, except that it should not precede the instruction address).
3. Instructions that operate on the accumulator have a blank operand field instead of "A".
4. When entering a branch instruction, following the branch mnemonic should be the **target** of the branch. If the destination address is not known at the time the instruction is entered, simply enter an address that is in the neighborhood, and later re-enter the branch instruction with the correct target address. **NOTE:** If a branch target is specified that is out of range, the mini-assembler will flag the address as being in error.
5. The operand field of an instruction can only be followed by a comment field, which starts with a semicolon (";"). Obviously, the Tiny assembler ignores the field and in fact will type over it when the line is typed over in disassembler format. This "feature" is included only to be compatible with future upgrades including input sources other than the keyboard.
6. Any page zero references will generate page zero instruction formats if such a mode exists. There is no way to force a page zero address to be two bytes, even if the address has leading zeroes.

In general, to specify an addressing type, simply enter it as it would be listed in the disassembly. For information on the disassembler, see the monitor section.

MONITOR ROM

Accessing the monitor

To enter the monitor from either BASIC type:

```
CALL -155 RETURN
```

To get the Apple to go into monitor mode upon RESET, do a CALL -155 to enter the monitor. After the monitor prompt (*), type:

```
3F2:69 FF 5A RETURN
```

and press RETURN. Then the Apple will go into monitor mode whenever you press RESET.

Autostart ROM reset vector

The Autostart ROM allows the machine language programmer to execute his own routine after a RESET instead of automatically dropping into the monitor or BASIC. The (new) Apple II Reference Manual describes how to use this feature on pages and .

CASSETTE INTERFACE

Cassette routines

Included is a description of the cassette tape routines in the Apple monitor and the Auto-Start ROM.

PRINTER INTERFACING

Controlling a printer from within a Basic program

The Apple has one output channel that can be directed to the screen or any of the peripheral slots (except slot 0). To change where the output goes use PR#n (where n is the slot number an 0 means the screen). Please note that PR# is also a DOS command and if you are using disk, it must be used in a PRINT with a control-D. Example:

```
10 PRINT "TO THE TV."  
20 PRINT CHR$(4);"PR#1"  
30 PRINT "TO THE PRINTER IN SLOT 1."  
40 PRINT CHR$(4);"PR#0"  
50 PRINT "BACK TO THE TV."  
60 END
```

Please read pages 100-102 of the DOS 3.2 manual for more information about PR # and DOS.

Centronics 730 normal/expanded printing

The Centronics 730 uses bit 8 to control the expanded print option but the Apple normally always sends that bit set, which means that the printer is always in expanded mode. I am including a program that allows software control of bit 8.

SOFTWARE ENTRY

First you must decide which slot the interface will go in and enter the driver. The routine is customized for this slot number and won't work properly if used with a different configuration. Enter the driver using the values from the table for words in brackets, < > .

SLOT	1	2	3	4	5	6	7
CODE	C1	C2	C3	C4	C5	C6	C7

Enter the monitor with CALL -155 and type

```

3B0:A9  <SLOT>
:20      95      FE
:A9      80
:20      ED      FD
:A9      C5
:85      36
:A9      03
:85      37
:4C      EA      03
:29      7F
:0D      CD      03
:4C      02 <CODE>
:80

```

To check your typing, type 3B0L and compare your listing to the one below for slot 1.

03B0-	A9	01		LDA	#\$101
03B2-	20	95	FE	JSR	\$FE95
03B5-	A9	80		LDA	#\$80
03B7-	20	ED	FD	JSR	\$FDED
03BA-	A9	C5		LDA	#\$C5
03BC-	85	36		STA	\$36
03BE-	A9	03		LDA	#\$03
03C0-	85	37		STA	\$37
03C2-	4C	EA	03	JMP	\$03EA
03C5-	29	7F		AND	#\$7F
03C7-	0D	CD	03	ORA	\$03CD
03CA-	4C	02	C1	JMP	\$C102
03CD-	80			???	

Now return to BASIC with 3D0G

SAVING THE PROGRAM TO DISK

The driver should be in memory before the printer is used. Save the driver by typing

```
BSAVE CEN 730, A$3B0, L$1E
```

USING THE PRINTER

The first time you want to use the printer you must load the driver and initialize the interface. From command mode type

```

BLOAD CEN 730
CALL 944

```

This may be done from a program by entering
100 PRINT D\$;"BLOAD CEN 730" : CALL 944
assuming that D\$ is a control-D.

If you want to switch back to the video monitor for output type
PR#0

or in a program enter
200 PRINT D\$;"PR#0"

Then to reconnect the printer, all that is required is CALL 954 or from a
program 300 CALL 954

SETTING THE PRINT MODES

To set normal print mode POKE 973,0

To set expanded print mode POKE 973,128

Tabbing on printer

Apple's interface cards are programmed to make nice listings of BASIC
programs. Unfortunately this feature interferes with the TAB function when
used with the printer. So instead of:

```
100 PRINT "HELLO"; TAB(70); "THERE!" use:
```

```
100 PRINT "HELLO";: POKE 36,70 : PRINT "THERE!"
```

You can follow the last quote with another ":" to continue tabbing.

6502 printing

To print characters to the Apple screen or other peripheral devices simply
requires sending the ASCII with the most significant bit set one character
at a time in the 6502 accumulator to \$FD18. Here is a short-cut if you
have ROM Applesoft.

```
300:LDA #$20
```

```
302:LDY #$03
```

```
304:JSR $DB3A
```

```
307:RTS
```

```
320:48 45 4C 4C 4F 00
```

```
  H E L L O
```

To change the slot you can load the accumulator with the slot number
and JSR to \$FD95 unless you are using DOS. With DOS you must print a
carriage return, control-D, "PR#", the slot number in ASCII and another
carriage return.

SERIAL CARD

Printer handshaking

The Apple High Speed Serial Interface card does not have built in hand-
shaking. This causes many printers to loose characters when running
faster than 300 baud. However, Diablo, Qume, or NEC Spinwriters can
get P8A PROM which will implement a software handshake with these
printers. I am enclosing information above using the data input line for
handshaking with other printers.

Line width 40 trashes program (Applesoft)

If you are using a printer with the line width set to greater than 40 and also sending the information to the Apple screen, you may wipe out your Applesoft program. Apple's screen is memory mapped and an off-set corresponding to a cursor position greater than 40 will cause the information to be "printed" in the same memory as the start of your Applesoft program. Therefore it is suggested that you use the printer or the video but not both at the same time.

COMMUNICATIONS CARD

Print routine fix for 3.2 or 3.2.1 DOS

The Communications Card does not generate a linefeed with a carriage return. On page 28 of the Comm Card Manual there is a "Printer Driver" which will supply linefeeds with carriage returns. Unfortunately the driver was written for DOS 3.1. To make the driver work with DOS 3.2 you need to enter the driver into memory and type:

34D:6E from the monitor or
POKE 845,110 from BASIC

then save and use the driver according to the directions on page 29.

Communications Card handshake

If you have a printer that requires handshaking you can have your Apple Communications Card modified by our service department to accept the proper signals. There is a service charge of \$35. Please contact the Apple Service Department for further details.

CTRL-R for remote terminal

To use your Apple and communications card from a remote terminal use the following steps:

On the Apple send: PR# <slot>
IN# <slot>

<Slot> is the number of the Apple I/O slot in which the Comm Card is inserted.

Or, if the Apple is acting as a "dumb terminal", from the external terminal send a CTRL R and a PR# <slot>.

To reverse this procedure:

From the terminal send: PR#0
IN#0

Or, to return the Apple to "dumb terminal" mode, send a CTRL T from the remote terminal.

How to use with a modem

Using the Comm Card with a modem is usually quite simple. Here is a quick example of how it's done . . .

Type: IN# <slot> using the slot number in which the Comm card is connected.

Then: CTRL A
CTRL F "Full Duplex" remote machine echos characters

Or: CTRL A
CTRL H "Half Duplex" Apple echos characters

The Com Card normally operates at 30 characters per second (300 Baud). If you are connecting to a machine that runs at only 10 characters per second (110 Baud) type:

CTRL A
CTRL 1 Set 10 CPS

Now turn on the modem, dial the remote computer and place the phone handset in the modem. You are now connected to the remote machine.

HIRES

SHAPE LOADER

The Applesoft SHLOAD statement loads shape tables from cassette tape and sets up the proper pointer so that Applesoft can use it. The following program loads the shape table from the disk as a binary file and sets up the proper pointer for Applesoft. The INPUT (line 100) and BLOAD (line 110 and 140) statements assume that the binary filename is prefixed by 'SHP.' because that is the format the SHAPE SAVE program on the next page uses. This routine should be the first thing in the program since it destroys the string variables in the process of loading the shape table.

```
100 INPUT "WHICH SHAPE TABLE TO LO AD ? SHP.";F$
110 PRINT CHR$(4);"BLOAD SHP.";F$;"A$D000"
120 DS = PEEK(977) + PEEK(978) * 256 + 3233
130 HM = PEEK(115) + PEEK(116) * 256 - PEEK(DS) -
    PEEK(DS + 1) * 256
140 PRINT CHR$(4);"BLOAD SHP.";F$;"A";HM
150 HIMEM: HM
160 POKE 232,HM - INT(HM / 256) * 256:
    POKE 233,HM / 256
```

SHAPE SAVER

This program saves your shape table on the disk as a binary file for the SHAPE LOADER routine to use. You should load this program before getting into the monitor and typing in your shape table. The best place to start your shape table is at \$2000. That leaves lots of room for the shape saver and the table. Use the instructions in chapter 9 of the Applesoft Reference manual. When you've got it all typed in and checked, type 3DOG to get back into Applesoft and RUN the program.

```
1000 REM SHAPE SAVER
1005 H$ = "0123456789ABCDEF"
1010 TEXT : HOME : VTAB 5
```

```

1020 INPUT "WHAT IS THE STARTING ADDRESS (HEX)?";A$
1030 IF LEN (A$) = 0 THEN END
1040 GOSUB 5000
1050 IF F < 0 THEN 1000
1060 S = X
1070 PRINT
1080 INPUT "WHAT IS THE ENDING ADDRESS (HEX)?";A$
1090 IF LEN (A$) = 0 THEN 1000
1100 GOSUB 5000
1110 IF F < 0 THEN 1000
1120 LET E = X
1130 IF E <= S THEN PRINT CHR$ (7);"THE END IS LESS THAN
    THE START!"; GOSUB 5100; GOTO 1000
1140 PRINT
1150 PRINT "WHAT NAME DO YOU WANT TO USE ?"
1160 INPUT "      SHP.";A$
1170 PRINT CHR$ (4);"BSAVE SHP.";A$;"A;S;"L";E - S + 1
1180 END
5000 PRINT : PRINT "CONVERTING TO DECIMAL"
5010 LET F = 0;X = F
5020 FOR J = 1 TO LEN (A$)
5030 FOR I = 1 TO 16
5040 IF MID$ (A$,J,1) = MID$ (H$,I,1) THEN X = X * 16 +
    I - 1; F = F + 1
5050 NEXT I,J
5060 IF F < > LEN (A$) THEN F = -1:"THAT'S NOT HEX!!!",
    CHR$ (7); GOSUB 5100
5070 IF X > 2 ^ 16 THEN F = -1: PRINT "THAT'S TOO BIG!!!";
    CHR$ (7); GOSUB 5100
5080 RETURN
5100 FOR J = 1 TO 1000; NEXT J; RETURN

```

Plotting on both screens

Plotting on both HIRES screens from Applesoft is relatively simple. The POKES listed on pages 132 and 133 of the Applesoft Reference Manual control which screen is displayed and location 230 (decimal) controls which screen will be modified by the HIRES commands. POKE 230,32 will cause Applesoft to draw on screen 1 (HGR) and POKE 230,64 will draw to screen 2 (HGR2).

RENUMBER PATCHES

Renumbers after an asterisk "*"

Renumber is a very powerful tool for developing programs but if you use it on you find some strange alterations in your program, Renumber may have done it. What happens is that the number after a "*" sometimes is mistaken as a line number and Renumber renumbers it. So if you had a line:

10 LET A = B * 10

it might renumber as:

20 LET A = B * 20

The fix is:

For RAM Applesoft

]LOAD RENUMBER

]POKE 14342,172

]POKE 14343,171

]SAVE RENUMBER

For ROM Applesoft

]LOAD RENUMBER

]POKE 4815,172

]POKE 4816,171

]SAVE RENUMBER

MISCELLANEOUS

PEEKs, POKEs, and CALLs

PEEK, POKE, and CALL all refer to machine language programs or data being used from a BASIC program. Often their use cannot be explained because a machine language program is loaded with or generated from the specific BASIC program. There is a list of general purpose PEEKs, POKEs, and CALLs in the (new) Applesoft II Reference Manual starting on page 128. The same information can be found in less convenient forms in the (new) Apple II Reference Manual.

CALLS - PEEKS, POKES & COLORS

Hex	BASIC Example	Description
Graphics Mode Controls		
C050:0	POKE-16304,0	Set color graphics mode.
C051:0	POKE-16303,0	Set text mode.
C052:0	POKE-16302,0	Clear mixed graphics.
C053:0	POKE-16301,0	Set mixed graphics (4 lines text).
C054:0	POKE-16300,0	Clear display Page 2 (BASIC commands use Page 1 only)
C055:0	POKE-16299,0	Set display to Page 2 (alternate).
C056:0	POKE-16298,0	Clear HIRES graphics mode.
C057:0	POKE-16297,0	Set HIRES graphics mode.
F836	CALL-1994	Clears upper 20 lines of text page 1 to reversed @ signs. If in low-resolution graphics mode, this clears the upper 40 lines of graphics screen to black (page 1 only).
F832	CALL-1998	Clears entire text page 1 to reversed @ signs. If in page 1 low-resolution graphics mode, this clears the entire screen to black.
F3F2	CALL 62450	Clears current high-resolution screen to black.
F3F6	CALL 62454	Clears current high-resolution screen to HCOLOR most recently HPLOTted. Must be preceded by a plot.

TEXT Mode Controls

FC58	CALL-936	(@ ^t) Home cursor, clear screen.
FC42	CALL-958	(F ^t) Clear from cursor to end of page.
FC9C	CALL-868	(E ^t) Clear from cursor to end of line.
FC66	CALL-922	(J ^C) Line feed.
FC70	CALL-912	Scroll up text one line.
0032:7F	POKE 50,127	Set inverse flag if 127.
0032:FF	POKE 50,255	Set normal flag if 255.

SCROLLING WINDOW

0020	POKE 32,L1	Set LEFT side of scrolling window to location specified by L1 in range of 0 to 39.
------	------------	---

0021	POKE 33,W1	Set window WIDTH to amount specified by $W1.L1 + W1 < 40$. $W1 > 0$.
0022	POKE 34,T1	Set window TOP to line specified by T1 in range of 0 to 23.
0023	POKE 35,B1	Set window BOTTOM to line specified by B1 in range of 0 to 23. $B1 > T1$.

CURSOR POSITION

0024	CH = PEEK(36) POKE 36,CH TAB(CH + 1)	Read/set cursor horizontal position in the range of 0 to 39. If using TAB, you must add "1" to cursor position read value; second and third example perform identical functions.
0025	CV = PEEK(37) POKE 37, CV VTAB(CV + 1)	Read/set cursor verticle position in the range of 0 to 23. If using TAB, you must add "1" to cursor position read value; second and third example perform identical functions.

SOUND

C030	X = PEEK(-16336)	Toggle speaker. (1 click in speaker)
C030:0	POKE-16336,0	
C020	X = PEEK(-16352)	Toggle cassette-output once (1 click on casette recording)

KEYBOARD

C000	X = PEEK(-16384)	Read keyboard; if $X > 127$ then key was pressed. (7F Hex)
C010:0	POKE-16368,0	Clear keyboard stobe - always after reading keyboard.

GAME PADDLES

C061	X = PEEK(-16287)	Read PDL (0) push button switch. If $X > 127$ then switch is "on." (7F Hex)
C062	X = PEEK(-16286)	Read PDL (1) push button switch. If $X > 127$ then switch is "on." (7F Hex)
C063	X = PEEK(-16285)	Read PDL (2) push button switch. If $X > 127$ then switch is "on." (7F Hex)
C058:1	POKE-16296,1	Clear Game I/O AN0 output. (OFF—3.5V high)

C059:0	POKE-16295,0	Set Game I/O AN0 output. (ON—,3V low)
C05A:1	POKE-16294,1	Clear Game I/O AN1 output. (OFF—3.5V high)
C05B:0	POKE-16293,0	Set Game I/O AN1 output. (ON—,3V low)
C05C:1	POKE-16292,1	Clear Game I/O AN2 output. (OFF—3.5V high)
C05D:0	POKE-16291,0	Set Game I/O AN2 output. (ON—,3V low)
C05E:1	POKE-16290,1	Clear Game I/O AN3 output. (OFF—3.5V high)
C05F:0	POKE-16289,0	Set Game I/O AN3 output. (ON—,3V low)

Colors As Set By Color

Note: Colors may vary depending on TV tint (hue) setting.

Low-Resolution Colors

0 = Black	8 = Brown
1 = Magenta	9 = Orange
2 = Dark Blue	10 = Grey
3 = Light Purple	11 = Pink
4 = Dark Green	12 = Green
5 = Grey	13 = Yellow
6 = Medium Blue	14 = Blue/Green
7 = Light Blue	15 = White

High-Resolution Colors

0 = Black 1
1 = Green
2 = Blue
3 = White 1
4 = Black 2
5 = (depends on TV)
6 = (depends on TV)
7 = White 2

ASCII CHARACTER CODES

DEC = ASCII decimal code HEX = ASCII hexadecimal code CHAR = ASCII character name n/a = not accessible directly from the APPLE II keyboard
CTRL = depression of control key while striking character indicated.

DEC	HEX	CHAR	WHAT TO TYPE	DEC	HEX	CHAR	WHAT TO TYPE
0	00	NULL	CTRL @	48	30	0	0
1	01	SOH	CTRL A	49	31	1	1
2	02	STX	CTRL B	50	32	2	2
3	03	ETX	CTRL C	51	33	3	3
4	04	ET	CTRL D	52	34	4	4
5	05	ENQ	CTRL E	53	35	5	5
6	06	ACK	CTRL F	54	36	6	6
7	07	BEL	CTRL G	55	37	7	7
8	08	BS	CTRL H or ←	56	38	8	8
9	09	HT	CTRL I	57	39	9	9
10	0A	LF	CTRL J	58	3A	:	:
11	0B	VT	CTRL K	59	3B	:	:
12	0C	FF	CTRL L	60	3C	<	<
13	0D	CR	CTRL M or RETURN	61	3D	=	=
14	0E	SO	CTRL N	62	3E	>	>
15	0F	SI	CTRL O	63	3F	?	?
16	10	DLE	CTRL P	64	40	@	@
17	11	DC1	CTRL Q	65	41	A	A
18	12	DC2	CTRL R	66	42	B	B
19	13	DC3	CTRL S	67	43	C	C
20	14	DC4	CTRL T	68	44	D	D
21	15	NAK	CTRL U or →	69	45	E	E
22	16	SYN	CTRL V	70	46	F	F
23	17	ETB	CTRL W	71	47	G	G
24	18	CAN	CTRL X	72	48	H	H
25	19	EM	CTRL Y	73	49	I	I
26	1A	SUB	CTRL Z	74	4A	J	J
27	1B	ESCAPE	ESC	75	4B	K	K
28	1C	FS	n/a	76	4C	L	L
29	1D	GS	CTRL shift-M	77	4D	M	M
30	1E	RS	CTRL ^	78	4E	N	N
31	1F	US	n/a	79	4F	O	O
32	20	SPACE	space	80	50	P	P
33	21	!	!	81	51	Q	Q
34	22	"	"	82	52	R	R
35	23	#	#	83	53	S	S
36	24	\$	\$	84	54	T	T
37	25	%	%	85	55	U	U
38	26	&	&	86	56	V	V
39	27	'	'	87	57	W	W
40	28	(	(	88	58	X	X
41	29	)	)	89	59	Y	Y
42	2A	*	*	90	5A	Z	Z
43	2B	+	+	91	5B	[	n/a
44	2C	,	,	92	5C	\	n/a
45	2D	—	—	93	5D	]	](shift-M)
46	2E	.	.	94	5E	^	^
47	2F	/	/	95	5F	_	n/a

TABLE OF THE ASCII CODES

CTRL	Character	Binary Bit 7 to Bit 0	Octal	Decimal	Hexadecimal	Character	Binary Bit 7 to Bit 0	Octal	Decimal	Hexadecimal	Character	Binary Bit 7 to Bit 0	Octal	Decimal	Hexadecimal
	@ NUL	00000000	000	000	00						V	01010110	126	086	56
A	SOH	00000001	001	001	01	+	00101011	053	043	2B	W	01010111	127	087	57
B	STX	00000010	002	002	02	-	00101100	054	044	2C	X	01011000	130	088	58
C	ETX	00000011	003	003	03	.	00101101	055	045	2D	Y	01011001	131	089	59
D	EOT	00000100	004	004	04	/	00101110	056	046	2E	Z	01011010	132	090	5A
E	ENQ	00000101	005	005	05	0	00101111	057	047	2F	[	01011011	133	091	5B
F	ACK	00000110	006	006	06	1	00110000	060	048	30	\	01011100	134	092	5C
G	BEL	00000111	007	007	07	2	00110001	061	049	31	]	01011101	135	093	5D
H	BS	00001000	010	008	08	3	00110010	062	050	32	^	01011110	136	094	5E
I	HT	00001001	011	009	09	4	00110011	063	051	33	_	01011111	137	095	5F
J	LF	00001010	012	010	0A	5	00110100	064	052	34	a	01100000	140	096	60
K	VT	00001011	013	011	0B	6	00110101	065	053	35	b	01100001	141	097	61
L	FF	00001100	014	012	0C	7	00110110	066	054	36	c	01100010	142	098	62
M	CR	00001101	015	013	0D	8	00110111	067	055	37	d	01100011	143	099	63
N	SO	00001110	016	014	0E	9	00111000	070	056	38	e	01100100	144	100	64
O	SI	00001111	017	015	0F	:	00111001	071	057	39	f	01100101	145	101	65
P	DLE	00010000	020	016	10	;	00111010	072	058	3A	g	01100110	146	102	66
Q	DC1	00010001	021	017	11	<	00111011	073	059	3B	h	01100111	147	103	67
R	DC2	00010010	022	018	12	=	00111100	074	060	3C	i	01101000	150	104	68
S	DC3	00010011	023	019	13	>	00111101	075	061	3D	j	01101001	151	105	69
T	DC4	00010100	024	020	14	?	00111110	076	062	3E	k	01101010	152	106	6A
U	NAK	00010101	025	021	15	@	00111111	077	063	3F	l	01101011	153	107	6B
V	SYN	00010110	026	022	16	A	01000000	100	064	40	m	01101100	154	108	6C
W	ETB	00010111	027	023	17	B	01000001	101	065	41	n	01101101	155	109	6D
X	CAN	00011000	030	024	18	C	01000010	102	066	42	o	01101110	156	110	6E
Y	EM	00011001	031	025	19	D	01000100	104	068	44	p	01101111	157	111	6F
Z	SUB	00011010	032	026	1A	E	01000101	105	069	45	q	01110000	160	112	70
[	ESC	00011011	033	027	1B	F	01000110	106	070	46	r	01110001	161	113	71
\	FS	00011100	034	028	1C	G	01000111	107	071	47	s	01110010	162	114	72
]	GS	00011101	035	029	1D	H	01001000	110	072	48	t	01110011	163	115	73
^	RS	00011110	036	030	1E	I	01001001	111	073	49	u	01110100	164	116	74
_	US	00011111	037	031	1F	J	01001010	112	074	4A	v	01110101	165	117	75
	SP	00100000	040	032	20	K	01001011	113	075	4B	w	01110110	166	118	76
!		00100001	041	033	21	L	01001100	114	076	4C	x	01110111	167	119	77
"		00100010	042	034	22	M	01001101	115	077	4D	y	01111000	170	120	78
#		00100011	043	035	23	N	01001110	116	078	4E	z	01111001	171	121	79
\$		00100100	044	036	24	O	01001111	117	079	4F	{	01111010	172	122	7A
%		00100101	045	037	25	P	01010000	120	080	50		01111011	173	123	7B
&		00100110	046	038	26	Q	01010001	121	081	51	~	01111100	174	124	7C
'		00100111	047	039	27	R	01010010	122	082	52	^	01111101	175	125	7D
(		00101000	050	040	28	S	01010011	123	083	53	~	01111110	176	126	7E
)		00101001	051	041	29	T	01010100	124	084	54	DEL	01111111	177	127	7F
*		00101010	052	042	2A	U	01010101	125	085	55					

CTRL	ABBR.	DESCRIPTION	CTRL	ABBR.	DESCRIPTION	CTRL	ABBR.	DESCRIPTION
@	NUL	— null, or all zeros	K	VT	— vertical tabulation	V	SYN	— synchronous idle
A	SOH	— start of heading	L	FF	— form feed	W	ETB	— end of transmission block
B	STX	— start of text	M	CR	— carriage return	X	CAN	— cancel
C	ETX	— end of text	N	SO	— shift out	Y	EM	— end of medium
D	EOT	— end of transmission	O	SI	— shift in	Z	SUB	— substitute
E	ENQ	— enquiry	P	DLE	— data link escape	[	ESC	— escape
F	ACK	— acknowledge	Q	DC1	— device control 1 (X ON)	\	FS	— file separator
G	BEL	— bell	R	DC2	— device control 2	]	GS	— group separator
H	BS	— backspace	S	DC3	— device control 3 (X OFF)	^	RS	— record separator
I	HT	— horizontal tabulation	T	DC4	— device control 4	_	US	— unit separator
J	LF	— line feed	U	NAK	— negative acknowledge		SP	— space
							DEL	— delete

OCTAL & HEX TO DECIMAL CONVERSION

DEC				DEC				DEC				DEC				DEC			
OCT	HX	times 256	DEC	OCT	HX	times 256	DEC	OCT	HX	times 256	DEC	OCT	HX	times 256	DEC	OCT	HX	times 256	DEC
0	0	0	0	164	34	13312	52	150	68	26624	104	234	9C	39936	156	320	D0	53248	208
1	1	256	1	165	35	13568	53	151	69	26880	105	235	9D	40192	157	321	D1	53504	209
2	2	512	2	166	36	13824	54	152	6A	27136	106	236	9E	40448	158	322	D2	53760	210
3	3	768	3	167	37	14080	55	153	6B	27392	107	237	9F	40704	159	323	D3	54016	211
4	4	1024	4	170	38	14336	56	154	6C	27648	108	240	A0	40960	160	324	D4	54272	212
5	5	1280	5	171	39	14592	57	155	6D	27904	109	241	A1	41216	161	325	D5	54528	213
6	6	1536	6	172	3A	14848	58	156	6E	28160	110	242	A2	41472	162	326	D6	54784	214
7	7	1792	7	173	3B	15104	59	157	6F	28416	111	243	A3	41728	163	327	D7	55040	215
10	8	2048	8	174	3C	15360	60	160	70	28672	112	244	A4	41984	164	330	D8	55296	216
11	9	2304	9	175	3D	15616	61	161	71	28928	113	245	A5	42240	165	331	D9	55552	217
12	A	2560	10	176	3E	15872	62	162	72	29184	114	246	A6	42496	166	332	DA	55808	218
13	B	2816	11	177	3F	16128	63	163	73	29440	115	247	A7	42752	167	333	DB	56064	219
14	C	3072	12	170	40	16384	64	164	74	29696	116	250	A8	43008	168	334	DC	56320	220
15	D	3328	13	101	41	16640	65	165	75	29952	117	251	A9	43264	169	335	DD	56576	221
16	E	3584	14	102	42	16896	66	166	76	30208	118	252	AA	43520	170	336	DE	56832	222
17	F	3840	15	103	43	17152	67	167	77	30464	119	253	AB	43776	171	337	DF	57088	223
20	10	4096	16	104	44	17408	68	170	78	30720	120	254	AC	44032	172	338	E0	57344	224
21	11	4352	17	105	45	17664	69	171	79	30976	121	255	AD	44288	173	341	E1	57600	225
22	12	4608	18	106	46	17920	70	172	7A	31232	122	256	AE	44544	174	342	E2	57856	226
23	13	4864	19	107	47	18176	71	173	7B	31488	123	257	AF	44800	175	343	E3	58112	227
24	14	5120	20	110	48	18432	72	174	7C	31744	124	260	BO	45056	176	344	E4	58368	228
25	15	5376	21	111	49	18688	73	175	7D	32000	125	261	B1	45312	177	345	E5	58624	229
26	16	5632	22	112	4A	18944	74	176	7E	32256	126	262	B2	45568	178	346	E6	58880	230
27	17	5888	23	113	4B	19200	75	177	7F	32512	127	263	B3	45824	179	347	E7	59136	231
30	18	6144	24	114	4C	19456	76	200	80	32768	128	264	B4	46080	180	350	E8	59392	232
31	19	6400	25	115	4D	19712	77	201	81	33024	129	265	B5	46336	181	351	E9	59648	233
32	1A	6656	26	116	4E	19968	78	202	82	33280	130	266	B6	46592	182	352	EA	59904	234
33	1B	6912	27	117	4F	20224	79	203	83	33536	131	267	B7	46848	183	353	EB	60160	235
34	1C	7168	28	120	50	20480	80	204	84	33792	132	270	B8	47104	184	354	EC	60416	236
35	1D	7424	29	121	51	20736	81	205	85	34048	133	271	B9	47360	185	355	ED	60672	237
36	1E	7680	30	122	52	20992	82	206	86	34304	134	272	BA	47616	186	356	EE	60928	238
37	1F	7936	31	123	53	21248	83	207	87	34560	135	273	BB	47872	187	357	EF	61184	239
40	20	8192	32	124	54	21504	84	210	88	34816	136	274	BC	48128	188	360	F0	61440	240
41	21	8448	33	125	55	21760	85	211	89	35072	137	275	BD	48384	189	361	F1	61696	241
42	22	8704	34	126	56	22016	86	212	8A	35328	138	276	BE	48640	190	362	F2	61952	242
43	23	8960	35	127	57	22272	87	213	8B	35584	139	277	BF	48896	191	363	F3	62208	243
44	24	9216	36	130	58	22528	88	214	8C	35840	140	300	C0	49152	192	364	F4	62464	244
45	25	9472	37	131	59	22784	89	215	8D	36096	141	301	C1	49408	193	365	F5	62720	245
46	26	9728	38	132	5A	23040	90	216	8E	36352	142	302	C2	49664	194	366	F6	62976	246
47	27	9984	39	133	5B	23296	91	217	8F	36608	143	303	C3	49920	195	367	F7	63232	247
50	28	10240	40	134	5C	23552	92	220	90	36864	144	304	C4	50176	196	370	F8	63488	248
51	29	10496	41	135	5D	23808	93	221	91	37120	145	305	C5	50432	197	371	F9	63744	249
52	2A	10752	42	136	5E	24064	94	222	92	37376	146	306	C6	50688	198	372	FA	64000	250
53	2B	11008	43	137	5F	24320	95	223	93	37632	147	307	C7	50944	199	373	FB	64256	251
54	2C	11264	44	140	60	24576	96	224	94	37888	148	310	C8	51200	200	374	FC	64512	252
55	2D	11520	45	141	61	24832	97	225	95	38144	149	311	C9	51456	201	375	FD	64768	253
56	2E	11776	46	142	62	25088	98	226	96	38400	150	312	CA	51712	202	376	FE	65024	254
57	2F	12032	47	143	63	25344	99	227	97	38656	151	313	CB	51968	203	377	FF	65280	255
60	30	12288	48	144	64	25600	100	230	98	38912	152	314	CC	52224	204				
61	31	12544	49	145	65	25856	101	231	99	39168	153	315	CD	52480	205				
62	32	12800	50	146	66	26112	102	232	9A	39424	154	316	CE	52736	206				
63	33	13056	51	147	67	26368	103	233	9B	39680	155	317	CF	52992	207				

